UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

TRABALHO DE CONCLUSÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Automação de uma Esteira Ergométrica para Aplicação em Realidade Virtual

Autor: Eduardo Varriale da Silva

Orientador: Marcelo Götz

Porto Alegre, dezembro de 17

Sumário

Agradecimentos

Primeiramente, a Deus, que me deu saúde para a realização deste trabalho e me abençoou com a vida, família, amigos e oportunidades que tenho.

À minha família, que sempre me apoiou incondicionalmente em todas as empreitadas da minha caminhada.

Ao Prof. Dr. Marcelo Götz, que me orientou ao longo deste projeto.

Ao meu tio Eng. André, que se empolgou no projeto comigo e me auxiliou com seus conhecimentos.

Aos meus avós, que, generosamente, me doaram sua esteira quando lhes contei minha ideia de trabalho de conclusão de curso.

Ao meu colega Richard, que prontamente me emprestou seu osciloscópio, que foi essencial em diversos momentos deste projeto.

A todos os professores, amigos e família que, direta ou indiretamente, fizeram parte da minha formação, o meu muito obrigado.

Resumo

Este trabalho de Conclusão em Engenharia de Controle e Automação tem por objetivo realizar a automação de uma esteira ergométrica. A velocidade da esteira é controlada de forma a manter o usuário sempre centralizado ao longo do eixo principal da mesma, permitindo-o alterar sua velocidade de caminhada livremente. Adicionalmente, é criado um ambiente em realidade virtual que emula ao usuário estar caminhando em um espaço aberto, mostrando simultaneamente as estatísticas de sua caminhada. Para tal, é utilizada uma esteira ergométrica comercial, que tem sua velocidade controlada por um microcontrolador, aproveitando parte do circuito de potência original da mesma. A posição do usuário é obtida por meio de um sensor ultrassônico instalado no painel da esteira e o ambiente virtual é criado para Google Cardboard, o qual utiliza um smartphone como base para o sistema de realidade virtual. A comunicação entre o smartphone e o microcontrolador ocorre por meio de rede Wi-Fi. Considerando as limitações da esteira e do sensor ultrassônico, cria-se um sistema que permite ao usuário uma melhor experiência durante o uso do equipamento, permitindo assim uma transição entre velocidades mais natural, já que a esteira acompanha o usuário e não o oposto.

Abstract

This final study on Control and Automation Engineering aims to perform the automation of an ergometric treadmill. The speed of the treadmill is controlled in order to always keep the user centralized along its main axis, allowing the user to vary their walking speed freely. Additionally, a virtual reality environment is created, emulating to the user the experience of walking in an open space, presenting simultaneously the statistics of the walk. In order to do that, a commercial ergometric treadmill, which has its speed controlled by a microcontroller, is used with part of its original power board. The user's position is acquired by an ultrasonic sensor installed on the monitor of the treadmill and the virtual environment is made for the Google Cardboard, which uses a smartphone as the basis for the virtual reality system. The communication between the smartphone and the microcontroller is established through a Wi-Fi network. Considering the limitations of both the treadmill and the ultrasonic sensor, a system is developed in which the user has a better experience during the usage, allowing a more natural transition in speed, as the treadmill keeps up with the user and not the other way around.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$u(t)$	Saída do controlador
$U(s)$	Transformada de Laplace de $u(t)$
K	Ganho proporcional do controlador
$e(t)$	Erro do controlador
$E(s)$	Transformada de Laplace de $e(t)$
T_i	Tempo integral do controlador
T_d	Tempo derivativo do controlador
$C(s)$	Função de Transferência do controlador
$G(s)$	Função de Transferência da planta
$H(s)$	Função de Transferência do sensor
$Y(s)$	Transformada de Laplace da saída da planta
$R(s)$	Transformada de Laplace da referência
$\Delta tEcho$	Tempo em que o pino e*cho* se manteve no nível alto
v_s	Velocidade do som no ar
$T(s)$	Função de Transferência do sistema em malha fechada a partir da referência
I_a	Corrente de armadura do motor
E_a	Tensão aplicada no motor
E_m	Tensão gerada pelo motor
R_a	Resistência de armadura do motor
V_e	Velocidade da esteira
C_e	Circunferência do rolo dianteiro da esteira
ΔT_{bd}	Tempo entre bordas de descida do sensor de velocidade
M	Massa da carga da esteira
F_a	Força de atrito entre a lona e a plataforma da esteira
T_m	Torque gerado pelo motor

T_r	Torque após a redução gerada pelas polias
C_m	Relação entre o giro do motor e a velocidade da esteira
g	Aceleração da gravidade
B	Coeficiente de atrito entre a lona e a plataforma da esteira
$\mathcal{L}\{f(t)\}$	Transformada de Laplace de $f(t)$.
L_a	Indutância da armadura do motor
k_{wi}	Constante de giro do motor para corrente de campo variável
i_f	Corrente de campo do motor
w	Giro do motor
k_{Ti}	Constante de torque do motor para corrente de campo variável
k_w	Constante de giro do motor para corrente de campo constante
k_T	Constante de torque do motor para corrente de campo constante
$P_u(t)$	Posição do usuário na esteira
$P_u(s)$	Transformada de Laplace de $P_u(t)$
t_s	Tempo de assentamento do sistema em malha fechada
p	Parcela real do polo dominante no sistema em malha fechada
$V_{ue}(s)$	Transformada de Laplace da velocidade do usuário
$T_p(s)$	Função de Transferência do sistema em malha fechada a partir da perturbação
z_2	Segundo zero do controlador PID
T_s	Tempo de amostragem
$C(z)$	Função de transferência discreta do controlador

Lista de Abreviaturas e Siglas

HMD	*Head-Mounted Display*
RV	Realidade Virtual
PWM	*Pulse-Width Modulation*
IGBT	*Insulated Gate Bipolar Transistor*
CC	Corrente Contínua
SDK	*Software Development Kit*
PID	Proporcional-Integral-Derivativo
fps	*Frames* por segundo

1 Introdução

Desde os anos 70, quando a indústria de jogos eletrônicos começou a crescer, até hoje, a tecnologia aplicada evoluiu muito no sentido de melhorar os gráficos e jogabilidade, para aumentar a imersão do jogador. Com o aumento do poder de processamento, gráficos com maior resolução e modelagens mais realistas foram possibilitados. A tecnologia da Realidade Virtual (RV) vem recebendo muita atenção na indústria dos jogos eletrônicos, com grandes empresas como Sony, Microsoft, Samsung e Facebook investindo no seu desenvolvimento. Ela permite inserir o usuário em um ambiente virtual, o qual ele pode explorar de maneira natural, olhando ao redor e se movimentando como em um ambiente real (Machado, 1995).

Um dos maiores desafios desta tecnologia é possibilitar interações do usuário com o ambiente de forma natural (Medina et al., 2008). Dentre estas interações, se locomover no ambiente é uma tarefa complicada, já que depende dos limites físicos do ambiente real, principalmente quando o ambiente virtual é maior que o real. As técnicas utilizadas para realizar essa movimentação podem ser divididas em duas categorias: as que tentam replicar a energia e o movimento do caminhar ou as que utilizam meios puramente virtuais. Dentre as técnicas consideradas puramente virtuais, incluem-se uso de "teleporte", apontar ou olhar na direção na qual se deseja andar e uso de controles direcionais (*joysticks*) (Zambaka et al., 2004). As técnicas que procuram replicar o movimento real incluem esteiras omni-direcionais, *motion foot pad*¸ ladrilhos robôs, sapatos acionados, *string walkers*, sistemas de "andar no lugar" e sistemas de rastreio de movimento em um ambiente real (Medina et al., 2008). Esta segunda categoria tende a impor no usuário um maior sentimento de presença e uma menor incidência de enjoo por conta do movimento (Zambaka et al., 2004).

Neste trabalho é analisada e implementada a técnica de movimentação por meio de uma esteira unidirecional. Ela permite, ao usuário, caminhar numa direção e ter seu movimento replicado no ambiente virtual. Para isso, é utilizada uma esteira ergométrica, controlada por um microcontrolador, que mantém o usuário no seu centro, ao longo do eixo de movimento. Isso permite que o usuário modifique sua velocidade e faz a esteira adequar sua velocidade a ele. O microcontrolador também envia a velocidade da esteira ao jogo de RV para que o jogo utilize a velocidade lida como a velocidade do jogador.

Para alcançar este objetivo, analisou-se o funcionamento e circuito da esteira ergométrica, modificou-se o circuito para possibilitar comandos externos, programou-se um microcontrolador para ler a distância do usuário, calcular a ação do controle e comandar a esteira, modelou-se o sistema, realizou-se a sintonia do controlador, desenvolveu-se um ambiente virtual e configurou-se a comunicação deste ambiente ao microcontrolador.

O restante do documento é dividido da maneira que segue. O capítulo seguinte é a revisão bibliográfica, onde se apresentam conceitos de controle e realidade virtual que serão úteis no decorrer do projeto. O capítulo 3 apresenta os materiais e métodos que serão utilizados durante o projeto e execução. O quarto capítulo trata do desenvolvimento e estudo de caso. Os capítulos posteriores apresentam os resultados, conclusões e trabalhos futuros do projeto.

2 Revisão Bibliográfica

2.1 Controle

Segundo Bazanella e Gomes da Silva, 2005, com base em ações de controle, que são operações matemáticas sobre o erro entre a variável manipulada (variável que se deseja manter igual ao valor de referência) e sua referência, controladores calculam um sinal que, aplicado ao processo, faz com que sejam satisfeitos determinados objetivos de desempenho do sistema de controle. O modelo de controlador mais utilizado em processos industriais é o PID que é formado das ações de controle proporcional, integrativa e derivativa, e daí vem seu nome. A ação proporcional confere uma reação direta ao erro presente no processo, fazendo o sistema reagir de forma imediata aos desvios da variável manipulada. A ação integral, utilizando o acumulado do erro ao longo do tempo, elimina o erro em regime permanente. A ação derivativa tenta antecipar o comportamento do processo, com base na variação do erro.

Um controlador PID gera um sinal que pode ser genericamente representado por:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \quad (1),$$

onde K é o ganho proporcional, $e(t)$ é o erro (diferença entre referência e variável controlada), T_i é o tempo integral e T_d é o tempo derivativo. A função de transferência para um controlador PID se encontra na Equação (2) e relaciona a saída de controle com a entrada do erro (Bazanella e Gomes da Silva, 2005).

$$C(s) = \frac{U(s)}{E(s)} = K\left(1 + \frac{1}{T_i s} + \frac{T_d N s}{s+N}\right) \quad (2)$$

Na Figura 1 pode ser observado um diagrama de blocos que representa um sistema controlado em malha fechada. Nela, C(s) é a transformada de Laplace do controlador, G(s) a transformada da planta a ser controlada e H(s) a do sensor. Para este sistema genérico, a função de transferência que pode ser obtida para a saída Y(s) a partir da R(s) é o mostrado na Equação (3).

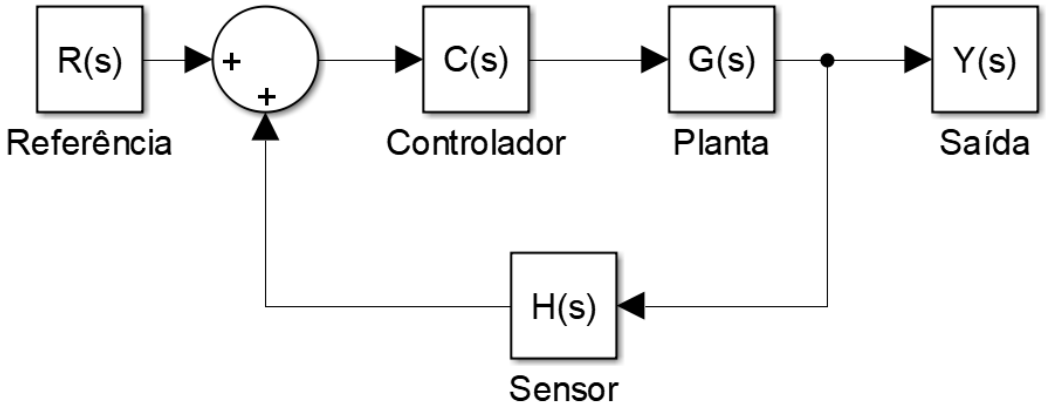


Figura 1 – Diagrama de blocos de um sistema controlado em malha fechada.

$$T(s) = \frac{Y(s)}{R(s)} = \frac{C(s)G(s)}{1 + C(s)G(s)} \tag{3}$$

Através do teorema do valor final pode-se obter facilmente o valor de uma variável no regime permanente a partir de sua transformada de Laplace multiplicada por s, com s tendendo a zero. Esse teorema está descrito matematicamente na Equação (4), considerando o sistema acima (Bazanella e Gomes da Silva, 2005).

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sY(s) = \lim_{s\to 0} sT(s)R(s) \tag{4}$$

2.2 Realidade Virtual

O termo Realidade Virtual é muito abrangente, e muitos autores não concordam totalmente com sua definição. Latta e Oberg, 1994, definiram o conceito de realidade virtual como uma avançada interface homem-máquina que simula um ambiente realístico e permite que participantes interajam com ele. Segundo Machado, 1995, dois fatores bastante importantes em sistemas de RV são imersão e interatividade. A imersão pelo seu poder de prender a atenção do usuário, e a interatividade no que diz respeito à comunicação usuário-sistema. Ainda segundo Machado, 1995, as tecnologias de RV começaram a aparecer com a Força Aérea dos Estados Unidos, após a segunda guerra, onde foram desenvolvidos simuladores de voo para o treinamento de pilotos. Os dispositivos para visão estereoscópica, muito utilizados hoje, começaram a ser utilizados na RV com o Sensorama (Figura 2), que é um equipamento patenteado em 1962 que procurava simular o ambiente por meio de diversos sentidos, com filmes 3D, som estéreo, aromas, ventiladores e vibrações mecânicas.

Figura 2 – Foto promocional de um protótipo do Sensorama. Fonte: Machado, 1995.

Também por volta dos anos 60, a Philco Corporation desenvolveu o primeiro videocapacete, também conhecido como *Head-Mounted-Display* (HMD), o Headsight (Scheinerman, 2009). O HMD é constituído de duas telas e um conjunto de lentes

especiais que ficam presas à cabeça do usuário, logo a frente de seus olhos. Este conjunto simula a visão estereoscópica, criando a ilusão de profundidade por apresentar aos olhos a imagem do objeto de duas perspectivas levemente diferentes. O videocapacete também possui sensores que medem a posição e orientação da cabeça, permitindo que a imagem transmitida se adapte à posição e perspectiva do usuário (Machado, 1995). Depois disso, muitas pesquisas e desenvolvimentos foram feitos, tanto por órgãos governamentais quanto por empresas do entretenimento, no sentido de aumentar o sentimento de imersão, novas formas de interação e de deixar a tecnologia mais acessível ao público.

Em 2014, a Google lançou o Google Cardboard, um óculos para realidade virtual que utiliza um visualizador simples, que pode ser feito até de papelão dobrado (exceto pelas lentes), como pode ser visto na Figura 3. Neste visualizador, o usuário insere seu celular com sistema Android ou iOS e a tela do celular e seus sensores, principalmente giroscópio e acelerômetro, são utilizados para criar a ilusão do ambiente virtual no usuário. Este visualizador torna a tecnologia muito acessível ao público geral, apesar de proporcionar uma experiência menos interativa do que concorrentes como o *HTC Vive* ou o *Oculus Rift* (Pierce, 2016). Com este produto, a Google tenta levar a realidade virtual ao maior número de pessoas, facilitando o desenvolvimento de aplicativos com suporte ao Cardboard e fazendo parcerias com empresas como o *The New York Times* para a distribuição de visualizadores. A Google disponibiliza um SDK (*software development kit*) para o motor gráfico Unity, tornando o desenvolvimento de aplicativos para o Cardboard simples para quem já sabe programar para o Unity.

Figura 3 – Google Cardbord aberto, com um *smartphone* inserido para visualização em RV. Fonte: Google, 2017.

Como dito por Aldaba et al., 2017, a realidade virtual sofre por possivelmente causar enjoo de simulação ou enjoo de movimento. O usuário pode desenvolver este mal-estar por conta de conflitos entre os sentidos que percebem os estímulos externos. Se a visão indica um movimento ou aceleração, mas o aparelho vestibular (também conhecido como órgão gravitoceptor) indica que está parada, este diferença pode causar enjoo, de maneira parecida com pessoas que se sentem mal por estar em um carro em movimento, focando o olhar em algo no interior do carro. Uma das maneiras de tentar mitigar este problema é simular no usuário da RV as sensações de caminhar.

Como citado em Medina et al., 2008, muitas empresas trabalham no desenvolvimento de diferentes plataformas para possibilitar esta movimentação no ambiente virtual, por meio de movimentos do usuário. Dentre elas, a Infinadeck (Figura 4), em um comunicado

de imprensa, informou ter um modelo comercialmente viável de sua esteira omnidirecional (Infinadeck, 2016). Trata-se de uma esteira movida por dois motores, que movem a lona em direções perpendiculares um ao outro. Isso permite que o usuário caminhe em qualquer direção, e a esteira se movimenta para mantê-lo no seu centro. Ela também possui um sistema de suporte que, preso à cintura do usuário, permite que o sistema capture seus pulos e agachamentos. A base do Infinadeck, onde se encontra a esteira, tem um comprimento do 1,7 m, largura de 1,6 m e altura de 0,4 m. Considerando a estrutura do suporte preso a cintura, o equipamento todo fica com mais de 2 m de altura.

Figura 4 – Usuário caminhando num ambiente virtual por meio do Infinadeck. Fonte: Infinadeck, 2016.

3 Materiais e Métodos

Para alcançar o objetivo de automatizar uma esteira ergométrica para controlar o ambiente virtual, serão utilizados uma esteira comercial, um sensor de distância ultrassônico, óculos de realidade virtual compatível com o Google Cardboard, um microcontrolador e o motor gráfico Unity.

3.1 Esteira Ergométrica

Como base para o projeto, utilizar-se-á uma esteira ergométrica da marca Athletic, modelo Trainee 4EE, de 2011, que foi escolhida por já estar disponível. Algumas das especificações técnicas da esteira, obtidas em seu manual de instruções (Athletic, 2011), estão na Tabela 1. O comprimento da face da lona sobre a qual o usuário caminha é de um metro e sua largura é 33 centímetros.

Tabela 1 – Especificações da Esteira Ergométrica

Motor	
Tipo	Corrente Contínua
Potência Máxima	1,4HP
Potência efetiva em esteira	1,0HP
Características	
Velocidade Mínima	1,0km/h
Velocidade Inicial	2,0km/h
Velocidade Máxima	8,0km/h
Usuário até	100 kg

Analisando-se a estrutura e circuitos da esteira (engenharia reversa), pôde-se melhor entender seu funcionamento. A vista explodida do modelo utilizado pode ser encontrada na Figura 5 e a legenda do índice das peças mais relevantes para este projeto é apresentada na Tabela 2, ambos retirados do manual de instruções da esteira (Athletic, 2011). Desmontando-se o monitor da esteira, encontra-se uma placa de circuito impresso. Esta placa é a responsável pelo cálculo e impressão na tela de LCD das informações como velocidade e distância percorrida. Para tal, ela está conectada ao sensor *Reed Switch*, que lê o número de voltas do rolo que move a lona. No circuito integrado também existem três botões, que são para modificar a informação mostrada no visor (chamado "*Mode*"), aumentar e diminuir a velocidade. Além disso, o monitor tem a chave de segurança que, se desconectada, para a esteira. Os botões de velocidade e a chave de segurança não são parte do circuito do visor, mas estão ligados, por um cabo multivias, à placa conversora.

Tabela 2 – Índices da Vista Explodida

Peça	Descrição
1	Placa conversora
2	Motor
3	Correia
4	Lona
5	Rolo dianteiro
6	Rolo traseiro

Peça	Descrição
7	Conjunto Plataforma + Perfil
8	Monitor
9	Chave de segurança
10	Cabo multivias
11	Cabo sensor de velocidade
12	Reed Switch

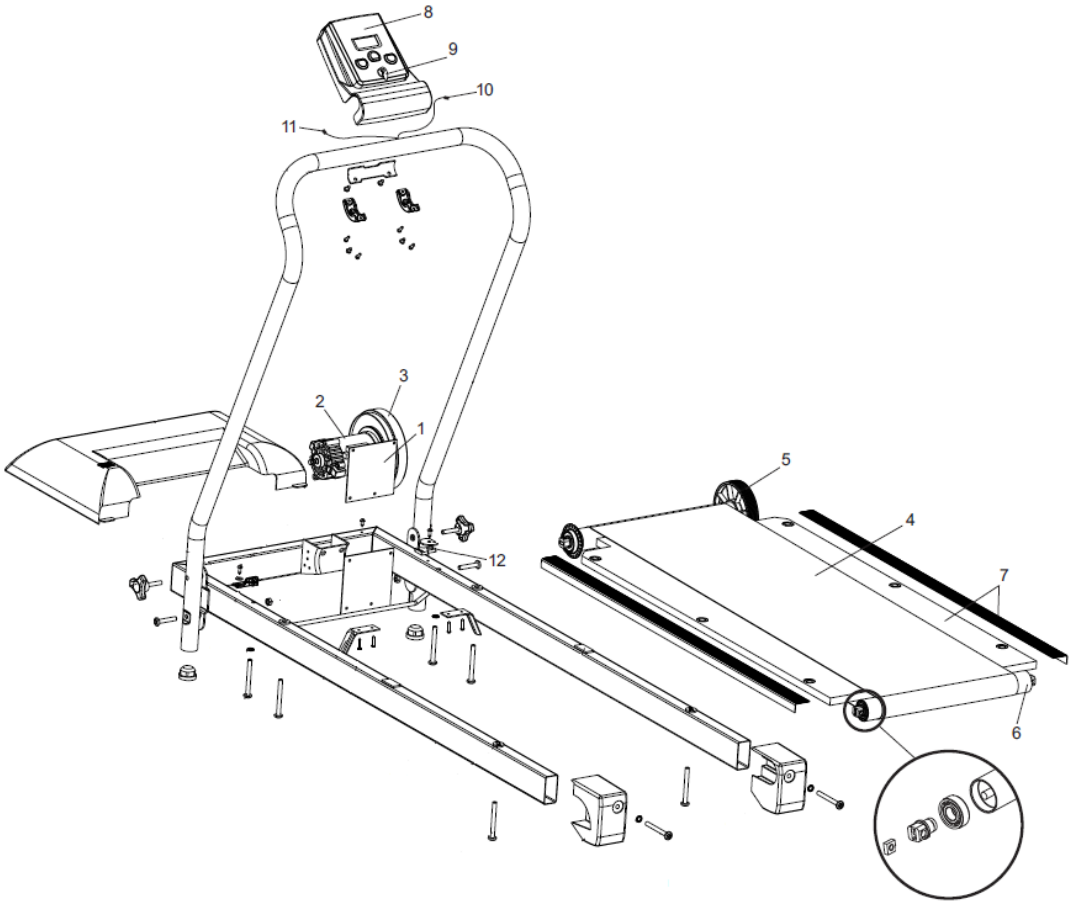


Figura 5 – Vista Explodida do modelo da esteira utilizado. Fonte: Adaptada de Athletic, 2011.

A placa conversora se encontra abaixo da carenagem da esteira, próxima ao motor e é ela quem possui todo o circuito de potência e toda a inteligência de controle de velocidade da esteira. Grande parte do seu circuito, principalmente a parte de controle, tem circuitos integrados que não possuem identificação, impossibilitando a análise de todo o circuito. A parte de potência do circuito pôde ser identificada analisando a placa, e um esquemático pode ser observado na Figura 6. Pode-se observar que o circuito entre a linha AC e a ponte retificadora é um circuito de segurança para reduzir efeitos de surtos na linha de alimentação e sobrecorrente. Após a ponte retificadora, existe um capacitor eletrolítico de 470 µF que filtra a tensão retificada, para ter uma tensão mais estável, de aproximadamente 175 V. Esta tensão é conectada ao motor CC pelo chaveamento do IGBT (IRGB20B60PD1), acionando o motor. Quando o IGBT não está saturado, mas funcionando como uma chave aberta, a corrente que a indutância do motor CC mantém passa pelo diodo D (8ETH06FP). O sistema de controle é alimentado por uma fonte de 15 V, gerada a partir de uma ponte retificadora de meia onda, resistores, capacitores e diodos zenner.

Utilizando um osciloscópio, pôde-se observar que durante o funcionamento da esteira, o sinal de controle que chega ao *gate* do IGBT é um PWM (*Pulse-Width Modulation*) com pulsos de aproximadamente 15 V e frequência 16 kHz. Segundo a ficha de dados da fabricante do IGBT (International Rectifier, 2003), a resposta do componente varia pouco para tensões entre *gate* e emissor de 12 V a 15 V.

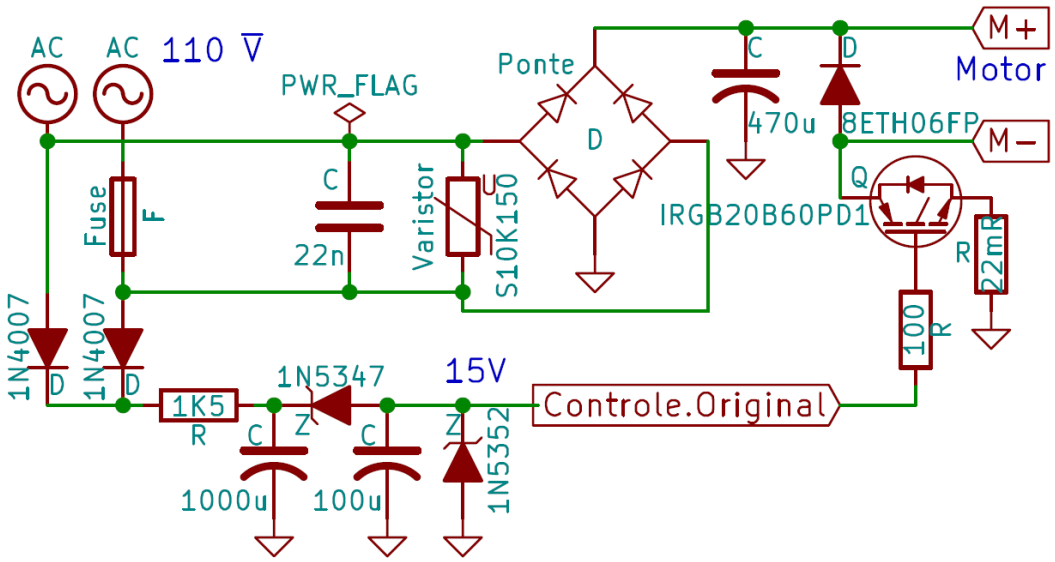


Figura 6 – Circuito de acionamento do motor da esteira

3.2 Microcontrolador

Na realização deste projeto, um microcontrolador é utilizado para comandar o motor da esteira, ler os sensores de posição do usuário e velocidade da esteira, calcular a ação de controle e comunicar com o ambiente virtual. Para tal fim, foi escolhido o microcontrolador *Wemos D1 R2* que é uma placa compatível com a IDE do Arduino e que oferece conectividade Wi-Fi nativa, por conta de ser baseada no controlador ESP8266EX. Ele possui 11 pinos I/O digitais e uma entrada analógica. Todas as suas portas funcionam com tensão máxima de 3,3 V, a alimentação pode ser feita por uma porta USB do tipo micro-B e ele opera com um *clock* de 80MHz (Wemos, 2017).

3.3 Google Cardboard

O Cardboard é um HMD (*head-mounted display*) simples anunciado pela Google na sua conferência Google I/O 2014. Ele foi criado com o intuito de ser o mais simples possível, para tornar a tecnologia de realidade virtual acessível para o maior número de pessoas. A Google liberou o design do seu produto e criou padrões para que outros fabricantes possam fazer o seu HMD compatível com o Cardboard (Coz et al., 2014). Neste projeto, será utilizado o modelo VRG-01, da marca Aquário. Ele é feito de material plástico, tem o suporte para celular e as lentes, mas não é totalmente compatível com as especificações do Google Cardboard por não possuir o botão magnético.

3.4 Unity

Unity é um motor de jogos que permite o desenvolvimento de aplicações para diversas plataformas (Unity, 2017). Ele oferece uma plataforma para criação de jogos e aplicativos em 2D, 3D, realidade virtual e realidade aumentada. Por meio do Unity Editor, podem ser criados jogos para diversas plataformas, como PC, dispositivos móveis, consoles e web, de forma simplificada por meio de interfaces gráficas e alguns modelos e funções prontas. No Unity, cada objeto da cena tem suas características e comportamentos definidos por “componentes”. Na biblioteca básica, já são oferecidos alguns componentes, mas *scripts* na linguagem C# de programação podem ser utilizados para criar componentes específicos de cada projeto (Unity, 2017). A biblioteca de desenvolvimento para Google Cardboard para Unity inclui uma câmera para o ambiente

virtual que estima a posição dos olhos e orientação da cabeça do usuário, corrige a distorção causada pelas lentes do HMD e mostra cena em visão estereoscópica (Coz et al., 2014).

3.5 Sensor ultrassônico

O HC-SR04 é um transdutor usado para medir distâncias através da contagem de tempo que uma onda de ultrassom leva para refletir no objeto e voltar ao sensor, como explicado em Nakatani et al., 2014. Ele consiste de um transmissor e um receptor ultrassônicos. Ele fornece medidas de 20 mm a 4000 mm, cuja precisão pode chegar a 3 mm (Nakatani et al., 2014). O HC-SR04 possui quatro pinos de entrada e saída: dois para alimentação 5 V (*Vcc* e *Ground*), um *trigger* e um *echo*.

Quando o *trigger* recebe um pulso de pelo menos 10 µs, o transmissor envia oito pulsos ultrassônicos de 40 kHz. O pino *echo* é colocado em nível alto durante o tempo que demorar entre o transmissor enviar o sinal e o sensor capta-lo. Em software, este tempo de nível alto do pino *echo* pode ser contado e utilizado para obter a distância segundo a Equação (5):

$$Distância = \frac{\Delta tEcho * v_s}{2} \quad (5),$$

onde $\Delta tEcho$ é o tempo em que o pino e*cho* se manteve no nível alto e v_s é a velocidade do som no ar. A divisão por 2 é necessária já que o som percorre duas vezes a distância, tendo que ir e voltar. Na Figura 7 encontra-se uma linha do tempo indicando o funcionamento do sensor ultrassônico.

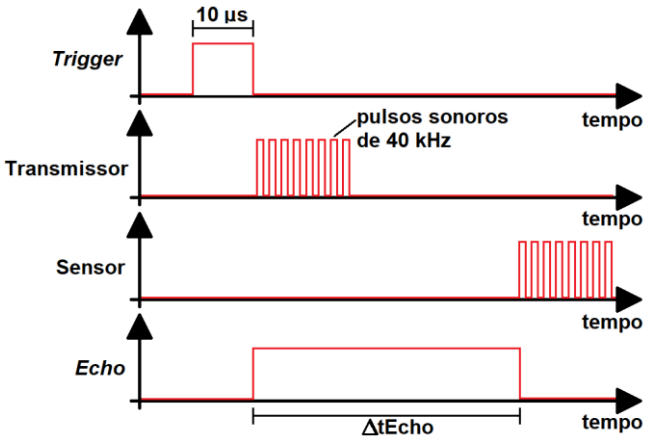


Figura 7 – Linha do tempo do funcionamento do sensor ultrassônico de distância.

3.6 Método do Lugar das Raízes

Segundo Bazanella e Gomes da Silva, 2005, o lugar das raízes é uma ferramenta gráfica que permite a visualização do efeito da mudança de um parâmetro K sobre os polos de uma função de transferência $T(s)$. Uma análise da posição dos polos da função de transferência permite a avaliação da resposta que este sistema terá quando receber uma entrada. Em Bazanella e Gomes da Silva, 2005, encontra-se uma explicação detalhada de como desenhar o lugar das raízes e como calcular os seus pontos mais relevantes.

4 Desenvolvimento e Estudo de Caso

Para facilitar o desenvolvimento do projeto, o mesmo é separado em seis partes: o microcontrolador, os circuitos de acionamento, a esteira, o sensor ultrassônico, o sensor de velocidade da esteira e o dispositivo móvel com a realidade virtual. Na Figura 8 podem ser observadas as relações entre as partes do projeto. Todas as comunicações entre as partes serão feitas por sinais em cabos, exceto a entre microcontrolador e dispositivo móvel, que é realizada por Wi-Fi.

Figura 8 – Integração entre as partes do projeto

4.1 Comando da esteira por microcontrolador

Como dito na Seção 3.1, a tensão no motor é regulada por meio de um sinal PWM com 15 V de tensão pico-a-pico que chega ao IGBT. Para comandar a esteira por meio do microcontrolador, o sinal PWM original é trocado pelo sinal vindo do microcontrolador, porém, como citado na Seção 3.2, a tensão de saída nas portas do microcontrolador é de 3,3 V, o que não saturaria o IGBT, possivelmente danificando-o. Projetou-se então, o circuito da Figura 9, onde o PWM do controlador é amplificado para um compatível com o IGBT. Neste circuito, a saída se manterá em 15 V enquanto o transistor NPN não estiver polarizado e em 0,2 V, quando o transistor estiver saturado. Por conta disso, o sinal PWM que o microcontrolador envia tem que ser o oposto do desejado no IGBT. Os resistores de 2k7 e 6k8 ohms garantem que o transistor não vai conduzir mesmo se a porta do microcontrolador esteja em aberta (como quando o microcontrolador está desligado), mas também formam um divisor de tensão que reduz a tensão no *gate* do transistor para 2,36 V. Esta redução não é significativa para o circuito já que a tensão necessária para

saturar o transistor é de menos de 1 V (ON Semiconductor, 2000). Adicionou-se um interruptor para que possa ser comutado entre o comando vindo do circuito original da esteira ou do microcontrolador.

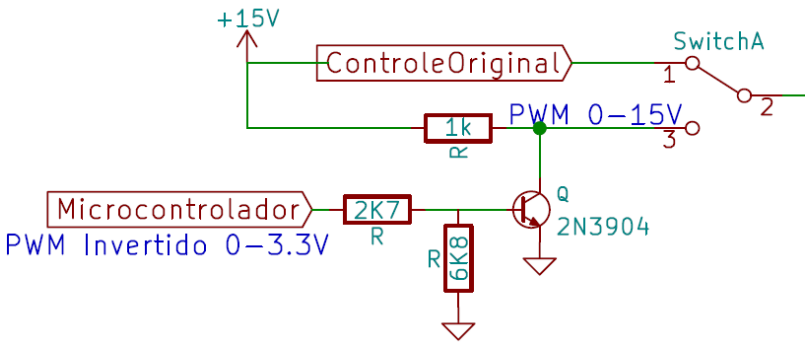


Figura 9 – Circuito de acionamento.

A programação do microcontrolador Wemos foi realizada na IDE do Arduino. Para o programa de geração do sinal PWM criou-se uma variável "vel" que é um inteiro que representa de zero a 1023 a razão cíclica do PWM. Utilizando a função *analogWrite*, o valor de "1023 – vel" é escrito na porta D3 do microcontrolador. Isso foi incorporado a um código de servidor Wi-Fi para interação com o microcontrolador que é tratado na Seção 4.4. A parte do código referente à escrita pode ser observada na Figura 10.

```
void loop(void) {
  unsigned long currentMillis = millis();
  float T = abs(currentMillis - previousMillis);
  if (T >= interval) //Só realiza a cada 'interval' milisegundos
  {
    //Lê distancia do usuário
    //Calcula Controle
    analogWrite(pinOut, 1023 - vel); //Escreve ação de controle no pino de saída
    previousMillis = currentMillis;
  }
}
```

Figura 10 – Código simplificado do *loop* do controlador.

A corrente no motor de corrente contínua, ignorando-se os efeitos do indutor, pode ser calculada pela Equação (6):

$$i_a = \frac{E_a - E_m}{R_a} \quad (6),$$

onde i_a é a corrente de armadura do motor, E_a é a tensão aplicada, E_m é o valor de tensão gerado pelo motor, que é proporcional à velocidade do mesmo e R_a é a resistência de armadura do motor (Bazanella e Gomes da Silva, 2005). Como a resistência de armadura deste motor é muito pequena (cerca de 2,4 Ω) e, quando parado, a E_m é zero, a tensão E_a não pode ser alta, ou a corrente se elevará muito, podendo queimar o IGBT ou outros componentes do sistema. Por outro lado, caso a tensão E_a não seja alta o suficiente para vencer a força de atrito e o motor se mantiver parado, o motor se portará como um curto e uma corrente muito alta passará por ele. Por isso, a velocidade é iniciada como zero. Depois, a tensão se eleva ao valor de 25,6 V (na forma de degrau de

tensão). Isto gera uma velocidade de aproximadamente 1,8 km/h para que o motor comece o giro e então o código impede que o valor diminua de 25,6 V para 0 V, para que o controlador não fique trocando entre os dois valores, gerando uma tensão média mais baixa e o motor não consiga girar, queimando o circuito de potência. Foi, também, estabelecida uma tensão média máxima de 60 V no motor, para limitar a velocidade da esteira a cerca de 4,5 km/h e evitar riscos ao usuário.

4.2 Sensor Ultrassônico

A Seção 3.5 detalha o funcionamento e a forma de utilização do sensor Ultrassônico. Para medir a distância ao obstáculo, deve ser aplicado um pulso de pelo menos 10 µs no pino *trigger* e então o sensor coloca o pino echo em nível alto durante o tempo que levar entre o envio e o recebimento das ondas. O microcontrolador deve, então, contar este tempo e calcular a distância a partir dele, utilizando a Equação (5),.

A programação no microcontrolador foi feita de forma a realizar a leitura da distância a cada 0,01 s, já que este é o tempo de ciclo escolhido para o controlador (Seção 4.6). Para tal, criaram-se as variáveis *currentMillis* e *previousMillis* que guardam respectivamente o tempo em milissegundos desde que o microcontrolador foi iniciado até o momento atual e o mesmo valor, mas para o loop anterior. *CurrentMillis* recebe seu valor a cada loop da função *millis()*, presente na biblioteca padrão do Arduino e *previousMillis* recebe o valor da variável *currentMillis* antes dela ser atualizada. Esta seção do código pode ser observada na Figura 10. Depois de atualizar estas variáveis, realiza-se um bloco *if* com a seguinte inequação como condição de entrada:

$$currentMillis - previousMillis \geq interval \quad (7).$$

Dentro desse bloco condicional ocorrem as medidas dos sensores e cálculos dos controladores. Criou-se a função *tempoUltrassom()*, que gera o pulso de 10 µs no pino *trigger* e retorna o tempo de nível alto do pino *echo*. A função *pulseIn*, nativa do Arduino, é utilizada para realizar a contagem, em microssegundos, do pulso no *echo*. Depois disso, realiza-se o cálculo da Equação (5), utilizando a velocidade do som em $m/\mu s$.

Durante testes realizados utilizando o sensor para medir a distância até uma folha de papel, observou-se que as medidas realizadas pelo sensor variam muito dependendo da inclinação entre folha e o sensor. Com a folha em paralelo ao sensor, o valor medido variava levemente em torno do valor real, porém, quando o obstáculo era inclinado, o valor lido aumentava significativamente. Para remediar este problema, realiza-se um tratamento no sinal lido. Primeiramente, limita-se o novo valor a ser no máximo 0,03 acima da posição anterior. Como a velocidade máxima da esteira são 2,22 m/s e a leitura é feita a cada 0,01 segundos, nunca deve ocorrer diferença maior de leitura. Caso a diferença seja maior, o novo valor é definido como o anterior mais 0,03. Depois, como a esteira utilizada possui um metro de comprimento, leituras acima de 110 cm são consideradas espúrias e ignoradas. Com as medidas que estão abaixo deste valor, o sinal é tratado com um filtro de média móvel com dez pontos. Este filtro modifica o valor do sinal para média entre os últimos 10 valores, suavizando derivadas e diminuindo o efeito de pequenas variações captadas pelo sensor.

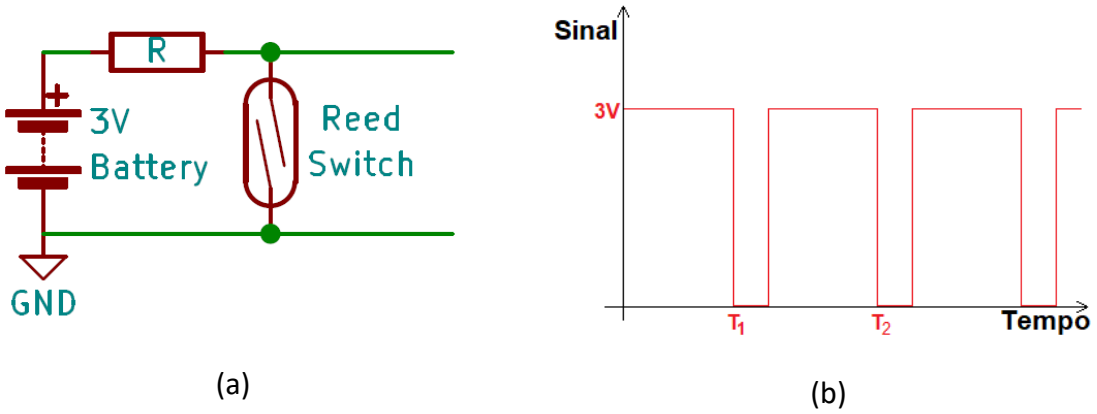
4.3 Sensor de Velocidade

Para a leitura da velocidade atual da esteira, que é mostrada no monitor original, faz-se uso de um sensor *Reed Switch*. Este sensor funciona como uma chave, que se fecha na presença de um campo magnético suficientemente forte. Conforme indicado com o índice 12, na Figura 5, o sensor está localizado ao lado do rolo dianteiro. Existe um imã fixado à roldana na extremidade do rolo dianteiro, de forma que a cada volta realizada pelo rolo, o *Reed Switch* fecha e abre seu contato uma vez.

Utilizando o circuito apresentado na Figura 11a, obtêm-se um sinal similar ao mostrado na Figura 11b nos terminais do *Reed Switch*. Como a frequência de chaveamento do sensor é a mesma do giro do rolo dianteiro, a velocidade com a qual a lona da esteira se move será proporcional ao tempo entre as bordas de descidas do sinal, ou seja, a $T_2 - T_1$. Esta relação é explicitada na equação:

$$V_e = \frac{C_e}{T_2 - T_1} = \frac{C_e}{\Delta T_{bd}} \quad (8),$$

onde V_e é a velocidade da esteira, C_e é a circunferência do rolo dianteiro e ΔT_{bd} é o tempo entre bordas de descida. No modelo de esteira utilizado, a circunferência C_e mede 0,14 m.



(a) (b)

Figura 11 – Sensor de Velocidade. a) Circuito de leitura do sensor; b) Sinal lido do sensor.

No Wemos, a leitura do sensor foi programada por meio de uma interrupção associada à borda de descida na porta do microcontrolador. Na função chamada pela interrupção, o valor de tempo, em milissegundos, desde a última instância da função é utilizado para calcular a velocidade pela Equação (8). Este tempo é obtido utilizando a função *millis()*, que é parte da biblioteca padrão do Arduino e retorna o tempo, em milissegundos, desde que a placa foi iniciada pela última vez. Durante a troca de estados, o sensor pode sofrer com *bouncing*, e por isso, se uma interrupção ocorrer menos de 63 ms depois da anterior, ela é ignorada, já que resultaria numa velocidade maior do que os 8 km/h máximos. Esta função pode ser encontrada na íntegra no Apêndice A: Programação do Microcontrolador, na função *speedInterrupt()*.

4.4 Servidor Wi-Fi

A configuração de um servidor Wi-Fi é simplificada pelo uso das bibliotecas disponibilizadas pelo fabricante do microcontrolador Wemos. Por meio das funções disponibilizadas, configurou-se a resposta do sensor a acessos ao IP do Wemos como um

texto que contém a saída do controlador, a distância do usuário lida e o valor da velocidade da esteira, lida pelo sensor de velocidade. O caminho "/ON" permite que o comando da esteira comece, enquanto "/OFF" desliga o comando e zera o controlador. Durante testes do sistema, outras funções foram criadas para que a largura do pulso pudesse ser modificada acessando-se caminhos no servidor. O caminho "/up" aumentava a largura em 1, "/10up" aumentava em 10 e suas contrapartes "/dw" e "/10dw" reduziam a largura da mesma forma. Pelo caminho "/SET", informava-se também o valor desejado para a largura do pulso em um parâmetro, modificando instantaneamente o valor para o desejado: "/SET?vel=80" levaria a largura do pulso para 80, por exemplo. Para realizar essa comunicação de forma prática, o IP da placa Wemos foi fixado no roteador, para que o mesmo não seja modificado a cada conexão à rede Wi-Fi.

4.5 Modelagem do Sistema

Representar um sistema de forma matemática é um passo importante para sua melhor compreensão e para possibilitar sua simulação. O modelo matemático que representa o sistema da esteira ergométrica será utilizado para sua simulação em ambiente computacional e para a criação de um controlador adequado para a finalidade do projeto.

O sistema estudado é formado, basicamente, do motor de corrente contínua, do sistema de redução, da lona da esteira e do usuário. Para representar o sistema, foi criado o diagrama da Figura 12. Nele, o usuário é representado pela massa M que, sobre a lona, gera uma força de atrito $\mathrm{F_a}$ entre a mesma e a plataforma da esteira (índice 7 da Figura 5). O torque gerado pelo motor é representado por $\mathrm{T_m}$, o torque após a redução gerada pelas polias é $\mathrm{T_r}$ e a velocidade da esteira é $\mathrm{V_e}$.

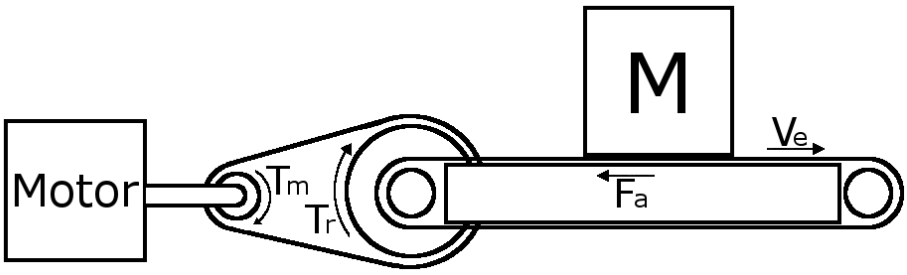


Figura 12 – Diagrama Mecânico.

Os raios das polias menor e maior são 0,85 cm e 6,5 cm, respectivamente, enquanto o do eixo dianteiro, solidário à polia maior, é de 2,23 cm. Utilizando estas informações e a segunda lei de Newton, pode-se chegar às equações:

$$C_m = \frac{0{,}065}{0{,}0085 \times 0{,}0223}\frac{1}{m} \quad (9),$$

$$M\frac{dV_e}{dt} = T_m C_m - MgB \quad (10),$$

$$\mathcal{L}\{V_e(t)\} = V_e(s) = \frac{T_m C_m - MgB}{Ms} \quad (11),$$

onde g é a aceleração da gravidade e B é o coeficiente de atrito entre a lona e a plataforma da esteira.

Como dito em Bazanella e Gomes da Silva, 2005, um motor de corrente contínua pode ser modelado a partir do circuito apresentado na Figura 13, com as equações:

$$L_a \frac{di_a}{dt} + R_a i_a + k_{wi} i_f w - E_a = 0 \quad (12),$$

$$T_e = k_{Ti} i_f i_a \quad (13).$$

Como o motor utilizado na esteira é de imã permanente, a corrente de campo será considerada constante. Dessa forma, o campo gerado pelo estator é constante, como no caso do imã permanente. As equações (14) e (15) unem $k_{wi} i_f$ na constante k_w e $k_{Ti} i_f$ na constante k_T.

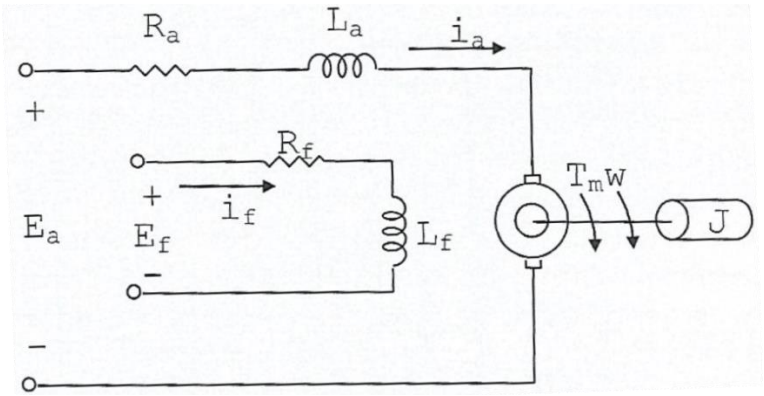


Figura 13 – Diagrama elétrico do motor CC. Fonte: Bazanella e Gomes da Silva, 2005.

$$L_a \frac{di_a}{dt} + R_a i_a + k_w w - E_a = 0 \quad (14)$$

$$T_m = k_T i_a \quad (15)$$

Utilizando a transformada de Laplace e unindo as equações (14) e (15), chega-se a Equação (16) para o torque T_m em função de E_a e do giro w.

$$T_m(s) = k_T \left(\frac{E_a - k_w w}{L_a s + R_a} \right) \quad (16)$$

Substituindo a Equação (16) na Equação (11), obtêm-se:

$$V_e(s) = \frac{k_t C_m}{M L_a s^2 + M R_a s + k_t k_w C_m^2} E_a(s) - \frac{MgB(L_a s + R_a)}{M L_a s^2 + M R_a s + k_t k_w C_m^2} \quad (17),$$

que descreve a velocidade desenvolvida pela esteira, V_e, em função da tensão aplicada no motor, E_a, e da carga sobre a esteira, M.

Para completar a modelagem e poder realizar simulações e ajustes de controlador com base no modelo, ainda é necessária a estimação dos parâmetros. C_m, g, e M já são conhecidos. C_m é a constante definida na Equação (9). g é a aceleração gravitacional,

considerada $9{,}81\ m/s^2$ neste projeto. M é a massa da carga, que é um parâmetro que varia a cada ensaio. R_a é a resistência da armadura do motor e pode ser facilmente obtida utilizando um ohmímetro nos terminais do motor. Para o motor deste sistema, a leitura de R_a é $2{,}4\ \Omega$. Os parâmetros k_t, k_w, L_a e B serão encontrados a partir de uma estimação de valores realizada com base em ensaios.

O modelo obtido para o sistema foi implementado em software no Simulink, resultando no diagrama de blocos da Figura 14. No sistema real, o coeficiente de atrito B seria separado em estático e cinético, com valor variável, impedindo que a velocidade se torne negativa. Porém, no modelo criado, o atrito é considerado sempre cinético e o bloco de saturação foi utilizado para impedir que a velocidade calculada se torne negativa, apenas permitindo resultados positivos quando a componente provinda do torque gerado pelo motor se torne maior que a provinda do atrito.

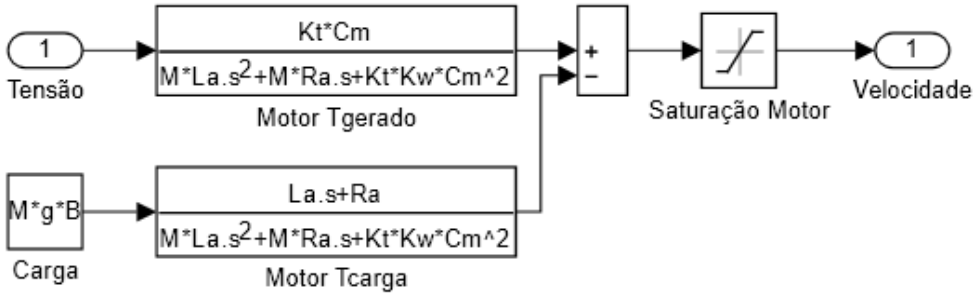


Figura 14 – Diagrama de blocos do modelo do motor.

Para a estimação dos parâmetros, realizaram-se dois ensaios no sistema, ambos com a mesma entrada de tensão no motor (sinal PWM que forma um pulso de valor médio de 25,6 V com duração de 10 s), mas um teste sem carga e outro com um usuário de 70 kg sobre a esteira. A leitura dos resultados é feita pelo *Reed Switch*. Nos ensaios, notou-se uma diminuição significativa na velocidade final da esteira com o aumento da carga.

Esses dados foram usados como entradas para a função *"Parameter Estimation"* do Simulink. Nela, os parâmetros k_t, k_w, L_a e B foram colocados para serem estimados e os dados dos ensaios fora inseridos como dois experimentos separados para a massa M poder ser alterada entre eles. Para o ensaio sem carga do usuário, a massa foi considerada 10 kg, representando as inércias do motor, polias e lona, sendo que erros nesta estimação serão minimizados com a estimação dos parâmetros. No ensaio com o usuário, a massa foi considerada 80 kg, sendo 10 kg das partes da esteira e 70 kg do usuário. Com estas configurações, a estimação de parâmetros minimiza o erro quadrático entre o resultado da simulação e o resultado dos experimentos, resultando nos valores da Tabela 3 para os parâmetros.

Tabela 3 – Parâmetros Estimados

Parâmetro	Valor Estimado
k_t	$0{,}026262\ NmA^{-1}$
L_a	$0{,}0011824\ H$

Parâmetro	Valor Estimado
k_w	$0{,}104289\ Vs\ rad^{-1}$
B	0.0374

Simulando o motor com o modelo obtido e com os parâmetros estimados, obtêm-se os resultados apresentados na Figura 15, juntamente com os resultados do experimento

real, para comparação. A resposta do sistema foi considerada adequada, principalmente para o caso com o usuário, que será a situação padrão do sistema. Para o caso sem a carga, o ganho estático se aproximou do valor desejado, mas a resposta dinâmica ficou mais rápida do que a do sistema real.

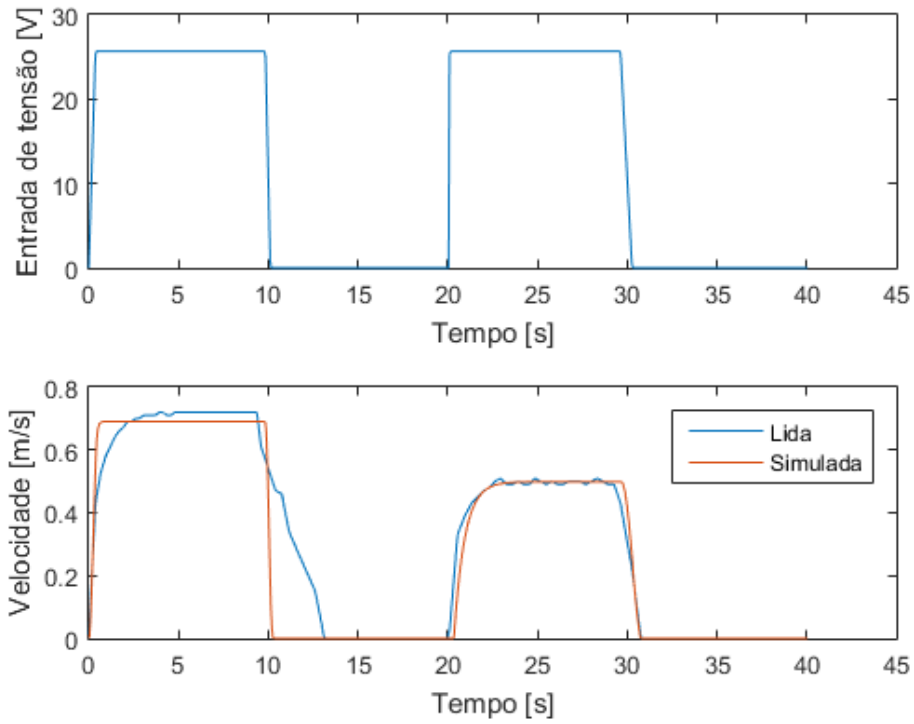


Figura 15 – Sinal aplicado, lido no ensaio do sistema e simulado.

Inserindo os valores dos parâmetros na Equação (17), considerando a massa da carga 80 kg, obtêm-se a equação:

$$V_e(s) = \frac{95.21}{s^2 + 2029{,}8\,s + 3404{,}8} E_a - \frac{0{,}367\,s + 744.71}{s^2 + 2029{,}8\,s + 3404{,}8} \quad (18).$$

A variável a ser controlada no sistema é a posição do usuário na esteira, não sua velocidade. A posição do usuário é a integral da velocidade relativa dele em relação ao corpo da esteira, como mostrado na Equação (19), onde $P_u(t)$ é a posição do usuário e V_{ue} é a velocidade relativa entre o usuário e a lona da esteira, ou seja, a velocidade com a qual o usuário corre. A Equação (20). mostra a transformada de Laplace da posição do usuário separada em dois termos. O primeiro relaciona a posição à tensão E_a, que pode ser manipulada. O segundo mostra a influência de variáveis que não podem ser manipuladas e que serão consideradas, portanto, perturbações.

$$P_u(t) = \int_0^t V_e(\tau) - V_{ue}(\tau) d\tau \quad (19),$$

$$P_u(s) = \frac{95{,}21}{s(s^2 + 2029{,}8\,s + 3404{,}8)} E_a(s) - \frac{1}{s}\left(\frac{0{,}367\,s + 744{,}71}{s^2 + 2029{,}8\,s + 3404{,}8} + V_{ue}(s)\right) \quad (20).$$

A implementação do sistema completo foi realizada no Simulink, e seu diagrama de blocos pode ser inspecionado na Figura 16. O subsistema chamado "Motor" é o apresentado na Figura 14, "Tensão" é a tensão E_a aplicada nos terminais do motor e "Usuário" é a velocidade do usuário relativa à lona, V_{ue}.

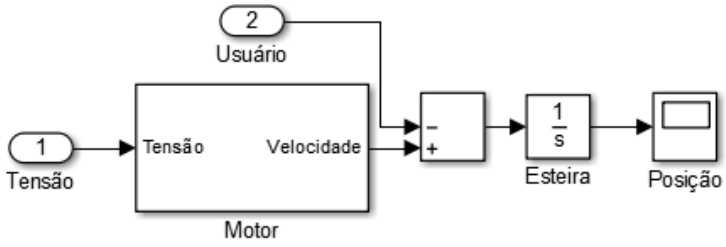


Figura 16 – Diagrama de blocos do modelo da esteira.

4.6 Projeto do Controlador

Tendo o modelo, podemos projetar um controlador baseado no modelo com o objetivo de acionar o motor para manter o usuário sempre centralizado na esteira, de forma que ele possa caminhar mais rapidamente ou mais devagar sem sair da lona. Para guiar o projeto do controlador, serão estabelecidas algumas especificações mínimas de desempenho do sistema realimentado. Deseja-se que o sistema responda com um tempo de assentamento menor que 1,5 s, com sobrepasso de 0%, sem erro no seguimento de referência em regime permanente e rejeição assintótica às perturbações. A variável controlada do nosso sistema é a posição do usuário, enquanto a manipulada é a tensão nos terminais do motor e as perturbações são a velocidade "retirada" pelo atrito e a velocidade do usuário. Ambas as perturbações serão consideradas constantes para os fins deste projeto do controlador. A referência é constante em 0,5 m, já que a esteira possui 1 m de comprimento e deseja-se manter o usuário no seu centro. Durante o projeto do controlador, será também assumido que o sensor de posição é ideal e pode ser representado simplesmente por um ganho unitário.

4.6.1 Controlador Analógico

A especificação de tempo de assentamento de 1,5 s indica que o sistema em malha fechada deve ter a parte real de todos os polos menor que -2,66, como indica a Equação (21), onde t_s é o tempo de acomodação e p é a parte real do polo dominante. Já a especificação de 0% de sobrepasso indica que todos os polos do sistema em malha fechada devem ser reais (Bazanella e Gomes da Silva, 2005).

$$t_s \approx \frac{4}{|p|} \tag{21}$$

Analisando a função de transferência relativa à tensão E_a, reescrita na Equação (22), pode-se observar que um dos seus polos tem parte real mais de dez oitavas acima dos outros. Sendo assim, seu efeito sobre a dinâmica da planta será muito pequeno, permitindo simplificá-la como a planta de segunda ordem descrita na Equação (23), onde se mantêm os polos dominantes e o ganho estático (Bazanella e Gomes da Silva, 2005).

$$P_u(s) = \frac{95.21}{s(s+1.679)(s+2028.12)} E_a \tag{22}$$

$$P_u(s) = \frac{0.04697}{s(s+1.679)} E_a \tag{23}$$

Pelo teorema do valor final, para termos o seguimento de referências do tipo salto com erro nulo em regime permanente, o modelo da referência deve aparecer no caminho direto e na malha de realimentação. Como a função de transferência da planta já possui um polo em zero, como o modelo do salto, o erro no seguimento de referência já vai ser nulo, mesmo apenas com um controlador do tipo proporcional. Entretanto, também pelo teorema do valor final, para haver rejeição assintótica da perturbação do tipo salto, o modelo da perturbação deve estar na malha de realimentação, mas não no caminho direto. Pela Figura 16 pode-se perceber que o polo em zero está também no caminho direto da perturbação, tornando necessária a existência deste polo também no controlador. Desta forma, o polo estará uma vez no caminho direto e duas vezes na malha de realimentação, garantindo a rejeição assintótica como mostrado na equação:

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sT_p(s)V_{ue}(s) = \lim_{s\to 0} s\frac{{}^{-1}/_{s}}{1+\frac{C'(s)G'(s)}{s^2}}\frac{1}{s} = \lim_{s\to 0}\frac{-s}{s^2 + C'(s)G'(s)} = 0 \tag{24},$$

onde $C'(s)$ e $G'(s)$ são o controlador e planta sem o seu polo em zero.

Desenhando o lugar das raízes (Figura 17) pode-se observar que um controlador puramente integrador tornaria o sistema instável para qualquer ganho, já que passam a haver polos com parte real positiva.

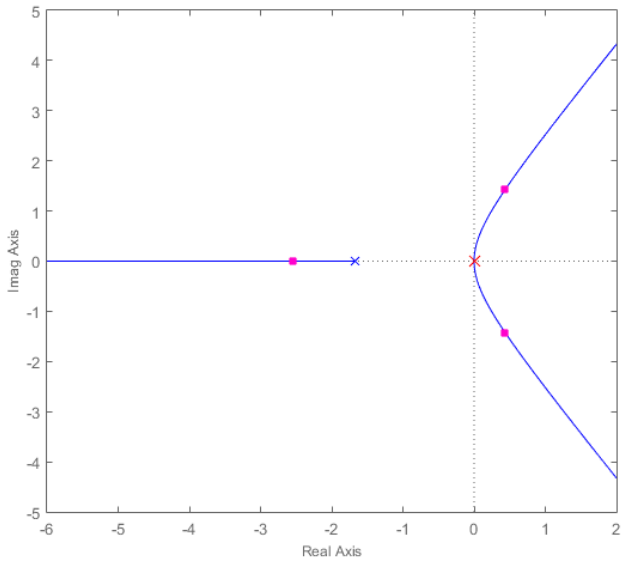


Figura 17 – Lugar das raízes do sistema em malha fechada com controlador puramente integral.

Pelas regas de construção do lugar das raízes é possível concluir que adicionando um zero ao controlador em alguns pontos entre 0 e -1,679 é possível ter todos os polos reais, respeitando a especificação de sobrepasso, mas esses polos seriam sempre limitados pelo valor de -1,679, se mantendo a sua direita, desrespeitando a especificação de tempo de acomodação. Conclui-se, portanto, que um controlador PI é insuficiente para as especificações desejadas. Na Figura 18 encontra-se o lugar das raízes obtido a partir de um controlador deste tipo.

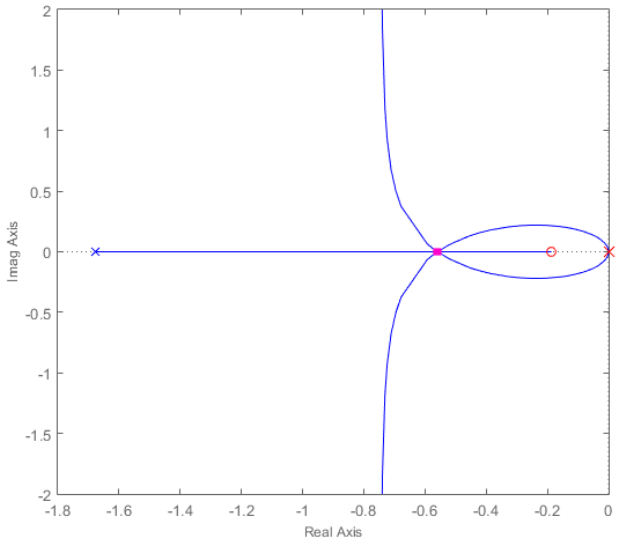


Figura 18 – Lugar das raízes do sistema em malha fechada com controlador PI.

Um controlador PID permite a escolha livre de mais um zero e implica também na existência de um polo com valor real de grande magnitude que tem pouca influência na resposta do controlador, mas garante a causalidade do mesmo. Utilizando um zero para cancelar o polo em -1,679, o ponto onde os polos voltam ao plano real não tem mais a limitação existente no controlador PI, permitindo levar este valor a pontos de parte real menor. Quanto maior for a magnitude deste ponto, porém, maior o ganho necessário para levar os polos a este ponto, aumentando o esforço de controle. Já que nosso sistema real possui limitações e pode saturar, ganhos grandes podem causar saturação e a consequente degradação do desempenho do controle. Para minimizar o ganho, respeitando as especificações, procurar-se-á colocar o zero no ponto exato de forma a fazer os polos voltarem ao eixo real em -2,66. Cancelar um polo da planta num sistema no qual as perturbações são um problema não é recomendado quando o polo está no caminho direto da perturbação, mas como o polo em -1,679 está apenas no laço de realimentação das perturbações, o cancelamento pode ser realizado. Estabelecendo este cancelamento e escolhendo o polo do derivativo em -300, temos o modelo do controlador como:

$$C(s) = K\frac{(s+1{,}679)(s+z_2)}{s(s+300)} \quad (25),$$

onde K e z_2 ainda devem ser escolhidos. Com esse controlador, a função de transferência do sistema em malha fechada é a descrita na Equação (26) e seus polos são ditados pela Equação (27).

$$T(s) = \frac{0.04697K(s+z_2)}{s^2(s+300)+0.04697K(s+z_2)} \quad (26)$$

$$s^2(s+300)+0.04697K(s+z_2) = 0 = 1+KF(s) = 1+K\frac{0.04697(s+z_2)}{s^2(s+300)} \quad (27)$$

Para o desenho do lugar das raízes, seguiram-se as instruções presentes em Bazanella e Gomes da Silva, 2005. Começando pela definição do número de assíntotas, ditados pela Equação (28) e o centro das assíntotas, pela Equação (29). Os pontos de entrada e saída

do eixo real são pontos pertencentes aos ramos sobre o eixo real do lugar das raízes que satisfazem a Equação (30).

$$nº\ polos\ de\ F(s) - nº\ zeros\ de\ F(s) = 3 - 1 = 2 \tag{28}$$

$$\frac{\sum polos\ de\ F(s) - \sum zeros\ de\ F(s)}{nº\ polos\ de\ F(s) - nº\ zeros\ de\ F(s)} = \frac{0 + 0 - 300 + z_2}{2} \approx -150 \tag{29}$$

$$\frac{d}{ds}\left(\frac{1}{F(s)}\right) = 0 \Leftrightarrow 0{,}09393s^2 + (14{,}091 + 0{,}14091z_2)s + 28{,}182z_2 = 0 \tag{30}$$

Como desejamos posicionar o zero z_2 de forma a um dos pontos de entrada e saída do eixo real ser em -2,66, podemos substituir os s da Equação (30) por esse valor e solucionar a equação para z_2. Com isso, encontra-se que:

$$z_2 \approx 1{,}3 \tag{31},$$

resulta na propriedade desejada.

K é o ganho que define em que ponto ao longo do lugar das raízes os polos da função de transferência em malha fechada se encontrarão. Desejamos que os polos se encontrem em -2,66 para satisfazer a exigência de desempenho estabelecida, portanto, substituindo na Equação (27) z_2 pelo valor estipula do na Equação (31) e s pelo valor desejado podemos obter o valor de K correspondente. Este valor se encontra na Equação (32) e a função de transferência completa do controlador, na Equação (33). A representação do lugar das raízes resultante é apresentada na Figura 19.

$$K = 32935 \tag{32}$$

$$C(s) = 32935\frac{(s + 1{,}679)(s + 1{,}3)}{s(s + 300)} = 326{,}15\left(1 + \frac{1}{1{,}3615s} + 0{,}333\frac{300s}{s + 300}\right) \tag{33}$$

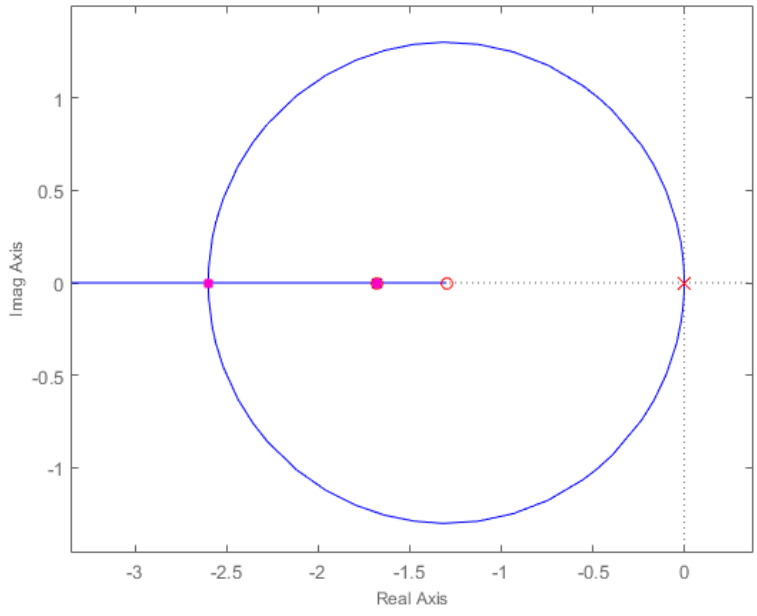


Figura 19 – Lugar das raízes resultante para o sistema com o controlador calculado.

Para testar este sistema de controle, o controlador foi adicionado à simulação, como mostrado na Figura 20. Aplicou-se então uma perturbação constante de 0,75 m/s e referência de 0,5 m. Em 20 s, após a estabilização do sistema, a referência sofreu um salto, passando a 0,6 m. A saída de posição frente ao salto na referência é mostrada na Figura 21. O tempo de assentamento gerado foi de 3,5 s, mas esta depreciação do desempenho do controlador já era esperada, já que o controlador e o atuador possuem limitações que não foram modeladas e são consideradas na simulação. O controlador foi configurado tendo saturações mínima e máxima em 25,6 V e 60 V respectivamente por serem os limites de tensão que podem ser aplicados no motor. O projeto realizado também considera a Equação Empírica (21), que foi criada para uma função de transferência com um único polo dominante e no processo de ajuste de controle, se colocaram dois polos na mesma posição.

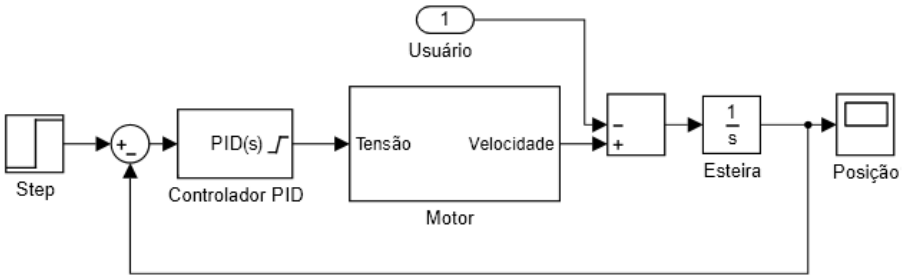


Figura 20 – Diagrama de blocos do sistema com controlador projetado.

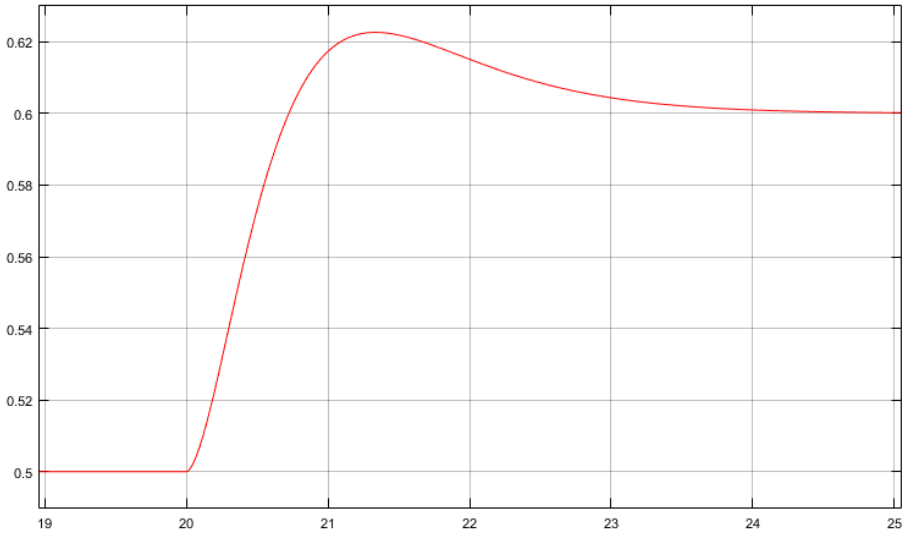


Figura 21 – Resposta ao salto na referência.

Para o projeto em questão, a referência sempre se manterá constante, tornando a principal função do controlador a rejeição às perturbações, sem que a posição do usuário extrapole os limites da esteira de 0 a 1 m. Na Figura 22 pode ser observada a variação abrupta de velocidades feita pelo usuário e seu efeito sobre a posição do mesmo na esteira. A simulação considera que, no início, o usuário acompanha a aceleração da esteira até sua velocidade mínima de 0,5 m/s. Depois, em 10 s aumenta sua velocidade para 1,15 m/s em um salto e em 20 s reduz para 0,75 m/s. O tempo para rejeição observado foi de 2,5 s e o valor da posição se manteve dentro dos limites da esteira.

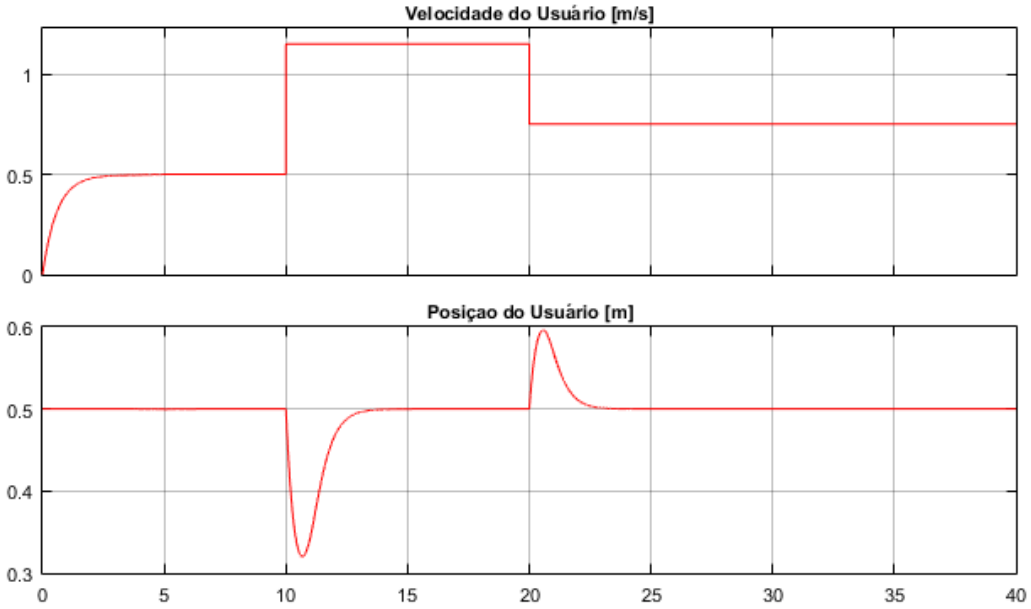


Figura 22 – Resposta da posição em relação a alterações na velocidade do usuário.

O controlador foi considerado suficiente para a aplicação, apesar das limitações do sistema de acionamento o deixar mais lento do que o planejado, porque rejeitou as perturbações assintoticamente e seguiu a referência.

4.6.2 Discretização do controlador analógico

O controlador criado na seção 4.6.1 é um controlador analógico, mas, no sistema final, deverá ser calculado por meio do microcontrolador, que é digital. Para tal, será realizada a discretização do controlador da Equação (33). A implementação digital será realizada por meio da aproximação de Tustin, que leva a função $C(s)$, no domínio de Laplace, para $C(z)$, no domínio da transformada Z (Bazanella e Gomes da Silva, 2005). A aproximação de Tustin é dada pela equação:

$$s \approx \frac{2}{T_s}\frac{z-1}{z+1} \quad (34),$$

onde T_s é o tempo de amostragem, que será escolhido como 0,01 s, já que é um tempo de ciclo que o microcontrolador consegue atender e é muito mais rápido do que as dinâmicas de controle que estamos interessados. Realizando a substituição da Equação (34) na Equação (33), obtêm-se a Equação (35) que representa a transformada Z do controlador discretizado.

$$C(z) = 326{,}15 + 1.1978\left(\frac{1+z^{-1}}{1-z^{-1}}\right) + 13033\left(\frac{1-z^{-1}}{1+0.2z^{-1}}\right) \quad (35)$$

A realização do controlador discretizado por um diagrama de blocos encontra-se na Figura 23. Nela, pode-se observar que as três ações de controle foram calculadas separadamente. Esta separação permite que um sistema *anti-windup* por integração condicional seja criado mais facilmente do que se a equação fosse implementada na sua forma simplificada. O sistema *anti-windup* impede que o controlador continue a integrar o erro quando a saída do controlador já se encontra saturada. Para a simulação do

sistema utilizando tempo discreto, colocou-se um bloco de atraso 0 no laço de realimentação para realizar a amostragem dos valores da posição.

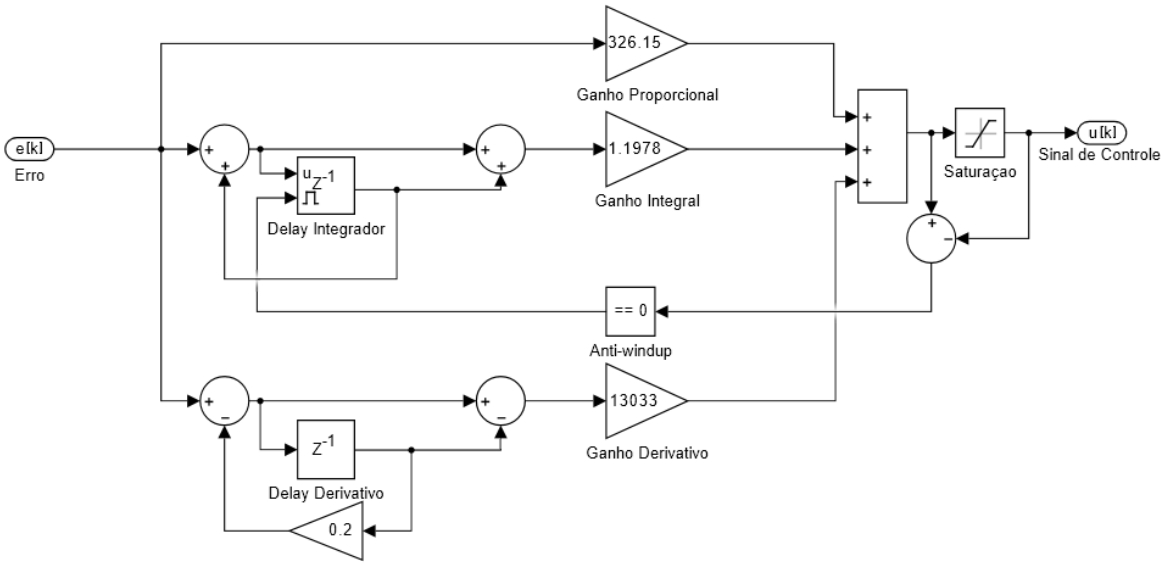


Figura 23 – Implementação do controlador com tempo discreto.

Realizando o mesmo teste de variação da velocidade do usuário que foi realizado no controlador analógico, obtêm-se resultados muito semelhantes e o tempo para rejeição observado se manteve em 2,5 s, como esperado. Os gráficos deste teste podem ser analisados na Figura 24.

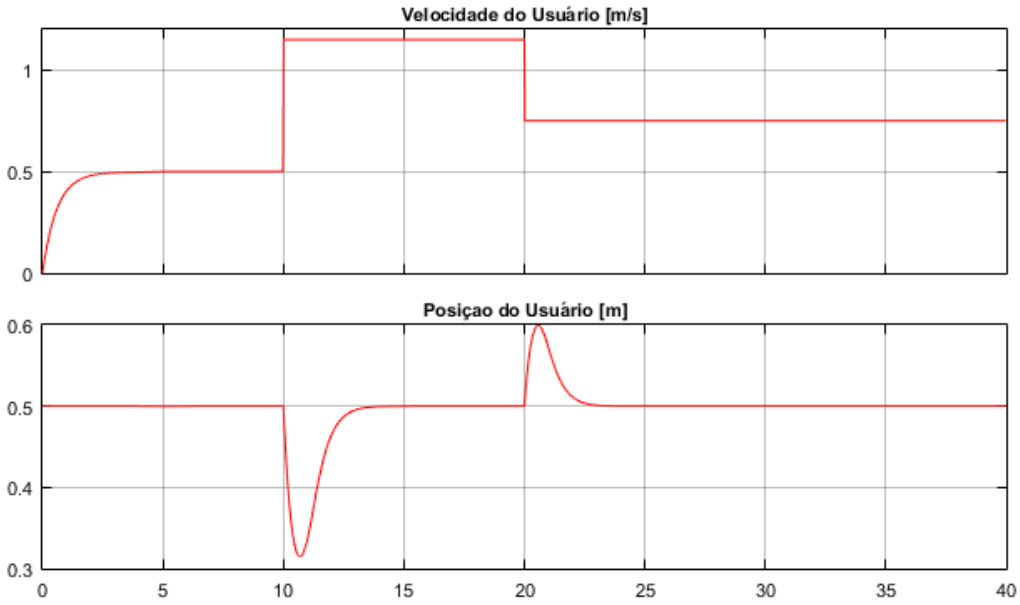


Figura 24 – Resposta da posição em relação a alterações na velocidade do usuário com controlador de tempo discreto.

A implementação deste controlador digital no microcontrolador pode ser analisada na função *loop()* do código no Apêndice A: Programação do Microcontrolador.

4.7 Ambiente Virtual

Sem a necessidade de um *feedback* visual de sua posição para ajustar sua velocidade, o usuário pode ser inserido num ambiente virtual. Este ambiente virtual poderia ser de qualquer tipo, como um jogo ou a recriação de um ambiente real, mas devem-se levar em consideração as limitações da esteira. Como a esteira apenas controla a posição do usuário em uma dimensão, o ambiente virtual deve permitir e incentivar a movimentação do usuário apenas nesta direção. Caso o usuário ande para os lados, o mesmo pode pisar fora da lona e cair.

Como a realidade virtual consome muito processamento do equipamento utilizado, o ambiente deve levar em considerações as limitações impostas pelo *hardware*. A tecnologia do Google Cardboard utiliza o processamento de um *smartphone*, diferente da utilizada pelo *Oculus Rift* ou pelo *HTC Vive*, limitando o número de polígonos utilizados na criação do ambiente para evitar a queda da taxa de *frames*, que é especialmente prejudicial na realidade virtual já que pode causar desconforto ao usuário (Nichols e Patel, 2002).

Com isso em mente, projetou-se um ambiente que simula uma caminhada tranquila em um parque japonês. Na Figura 25 pode-se observar como o ambiente se parece quando em uso. No ambiente, o terreno é um gramado com um caminho de areia bem definido, para indicar ao usuário que ele deve andar nesta direção. Este caminho é alinhado com a direção da esteira, para que o usuário não saia da lona. Nas laterais do caminho, existem árvores similares a cerejeiras, com tronco escuro e floração rosada. As árvores foram criadas a partir de um componente do Unity chamado "*Tree*" que permite a criação de árvores pela adição de ramos e folhas num diagrama de hierarquia. Junto às árvores, colocaram-se emissores de partículas que criam pétalas caindo das árvores. Estas pétalas adicionam dinamismo ao ambiente que sem elas tem apenas objetos estáticos.

Figura 25 – Imagem do ambiente de realidade virtual criado

Ao longo do caminho, a cada 50 m, encontra-se um *torii* ou um *tōrō*, alternadamente, dando ao usuário pontos de referência nos quais pode visualizar a distância percorrida. *Torii* são portões tradicionais japoneses que são utilizados em caminhos para indicar crescente proximidade a um local sagrado e *tōrō* são lanternas de pedra tradicionalmente utilizadas para iluminar o caminho a um templo (Prohl e Nelson, 2012). O céu foi adaptado do pacote de *skyboxes* "*Sky5x One*" disponível gratuitamente na "*Asset Store*" do Unity. Ao céu que vem no pacote, foi adicionada a sombra de um castelo japonês no horizonte, para dar ao usuário um destino ao qual caminhar.

O projeto no Unity contém quatro objetos principais: a "*MainCamera*", o "Cenário", a "*Boundary*" e a "UI". Cada um destes objetos possuem outros objetos filhos e componentes que são associados a eles. Na Figura 26 está a vista superior do projeto, com algumas indicações que serão úteis para a explicação do funcionamento do ambiente.

A "*MainCamera*" é um objeto padrão da biblioteca do Google Cardboard. Ela simula a câmera estérea necessária para a geração da imagem 3D e já possui os *scripts* básicos de interação com o usuário, como o *"RayCaster*" e o seguimento da orientação da cabeça do usuário segundo a leitura dos sensores do celular. O *"RayCaster*" é um mecanismo que indica a outros objetos quando eles estão no centro do olhar do usuário, permitindo interações com os mesmos.

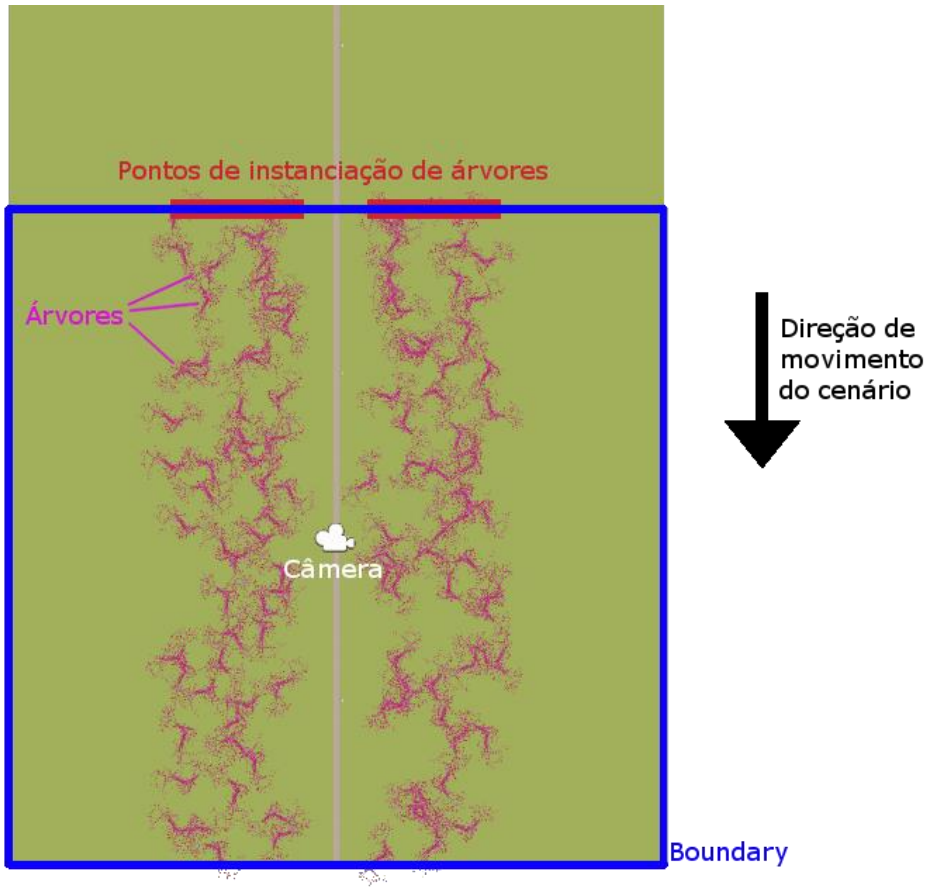


Figura 26 – Vista superior do ambiente virtual

O "Cenário" tem como objetos filhos dois terrenos quadrados de 200 m cada, as árvores e as decorações. Dois scripts foram criados como componentes para o cenário:

"Cenary Speed" e "Spawn Cenary". O primeiro acessa o IP do microcontrolador a cada 0,1 s para receber por ele o valor de velocidade lido do *Reed Switch*. Ele aplica, então, este valor como velocidade de todo cenário, portanto, quando o usuário anda na esteira, o cenário se move com a mesma velocidade sob a câmera no ambiente virtual. O segundo script é o que instancia e adiciona as árvores ao cenário, conforme o usuário anda por ele. Ele insere duas árvores por vez, uma de cada lado da trilha, sendo que sua distância da trilha e rotação no eixo vertical são valores aleatórios, dentro de limites. A região onde o instancimento de árvores pode ocorrer está indicada na Figura 26. A distância entre árvores na direção da movimentação do cenário é definida por uma espera entre a as inserções, e esta espera também é definida por uma função aleatória limitada e leva em consideração a velocidade do cenário. Ambos os scripts podem ser mais analisados no

Apêndice B: Programação de Scripts para Unity.

A "Boundary" é um cubo invisível de lado 200 m em torno do usuário, indicada em azul na Figura 26. Todos os objetos do cenário, quando saem de dentro da "Boundary", ativam o script "DestroyOnExit". Quando uma árvore ativa esta função ela é destruída, e quando o terreno ou as decorações ativam esta função, eles são transportados 400 m para frente. Desta forma, cria-se a ilusão de um caminho infinito, porém sem gastar processamento mantendo objetos que já estão distantes do usuário, reutilizando o terreno e decorações e apenas criando novas as árvores.

A "UI" é a interface com o usuário. Composta de três botões posicionados no ambiente virtual, ela tem a função de iniciar, parar e mostras estatísticas da corrida ao usuário. Os botões são ativados após manter-se o olhar sobre eles durante um segundo. Esta função é parte da biblioteca do Google Cardboard. Na Figura 27 é possível ver os três botões. A imagem foi tratada para ofuscar o resto do cenário e destacar a interface. O botão "Go!" manda ao microcontrolador, via Wi-fi, o sinal para iniciar o programa de controle de posição e indica ao script "Cenary Speed" que deve começar a leitura de velocidade. O botão "Stop" faz o inverso, mandando os sistemas pararem. Esses dois botões desaparecem quando são ativados e fazem com que o outro apareça. Desta forma, apenas um deles aparece ao usuário por vez. O botão "Stats" faz com que a janela de estatísticas, que mostra a velocidade e distância percorrida, apareça e desapareça.

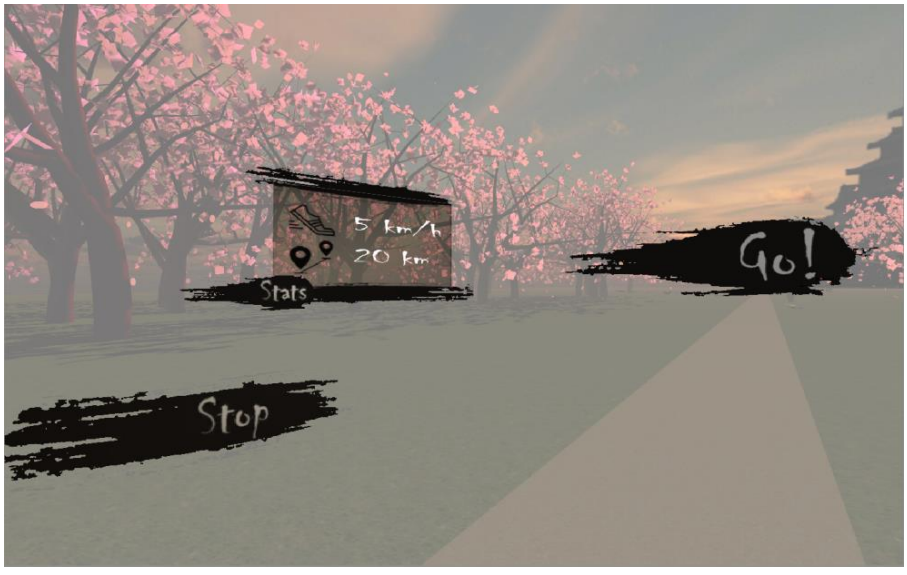


Figura 27 – Botões da interface com o usuário

5 Resultados

Analisando um dos testes realizados, cujos dados são apresentados nas figuras Figura 28, Figura 29 e Figura 30, observa-se que a distância do usuário da parte frontal da esteira se mantém entre 0,35 e 0,7, próxima de sua referência de 0,5. Neste teste, o usuário começa fora da esteira, e sobe na mesma no tempo 5 s. Quando em cima da esteira, o usuário aumenta e reduz sua velocidade duas vezes e depois, em 105 s, desce da esteira. Na Figura 28, em azul claro, está indicada a distância lida pelo sensor ultrassônico, antes da filtragem dos dados, em metros. Em azul escuro, as leituras já filtradas, também em metros, e, em vermelho, a velocidade lida a partir do *Reed Switch*, em metros por segundo. O eixo das abcissas é o tempo, em segundos. Observa-se que no período, a velocidade variou entre 0,55 e 1,25 m/s. Durante as leituras, o sensor de velocidade apresenta alguns valores espúrios, muito mais altos ou baixos que seus vizinhos, que são causados pelos *bounces* do sensor que não foram filtrados. Estes valores não chegam a afetar a simulação da realidade virtual, por serem poucos e não serem consecutivos. Pelos dados de distância não filtrados observa-se a imprecisão do sensor e a importância da filtragem.

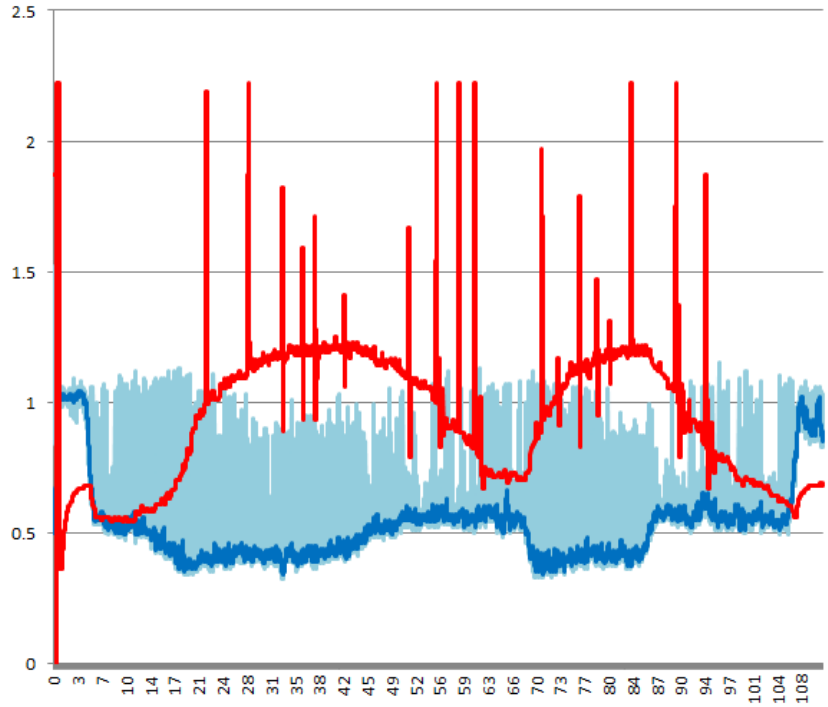


Figura 28 – Posição lida pelo sensor (azul claro), posição filtrada pelo microcontrolador (azul escuro) e velocidade da esteira lida (vermelho).

Na Figura 29, é mostrada em azul a saída do controlador em Volts e, em vermelho, a saída quando aplicado um filtro de média móvel central, utilizando os valores desde um quarto de segundo antes a um quarto de segundo depois do ponto. Este sensor de média móvel foi aplicado durante a análise posterior dos dados, e não é realizada durante o funcionamento do projeto. Pela Figura 30, que apresenta as saídas individuais de cada parcela do controlador, em verde a parcela proporcional, em vermelho a integrativa e em azul a derivativa, todos em Volts, observa-se que a parcela derivativa é a responsável por grande parte da alta variância do controlador. Esta variância da saída do controlador,

porém, é filtrada pela lenta dinâmica do controlador, resultando na curva de velocidade suave mostrada na Figura 28.

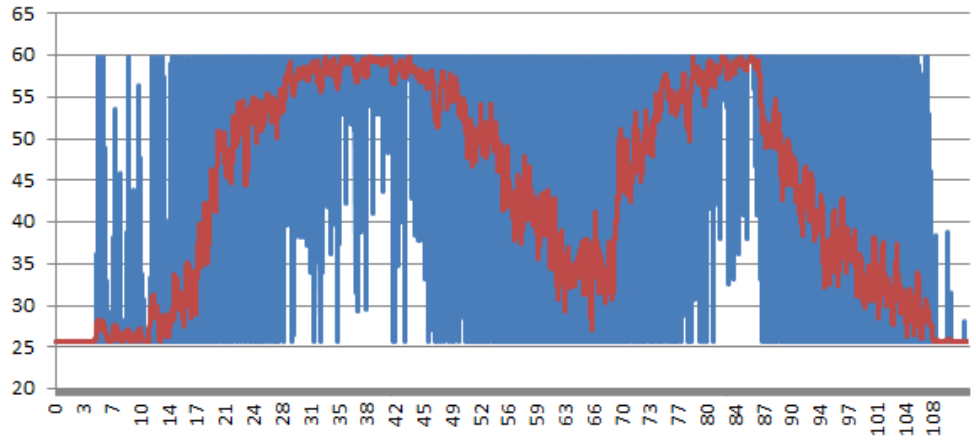


Figura 29 – Saída do controlador como durante o funcionamento e filtrada.

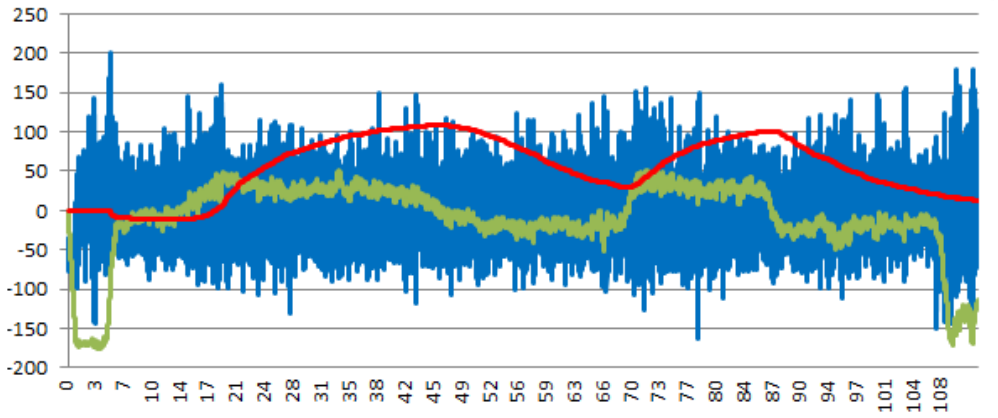


Figura 30 – Saída individual de cada parcela do controlador.

Em relação à realidade virtual, o projeto seguiu o funcionamento esperado. A esteira ligou quando o botão virtual de partida foi acionado e parou com o botão de parada. O sentimento de imersão do usuário foi considerado bom e o usuário não apresentou enjoo durante o uso. Este uso não foi durante um tempo prolongado ininterrupto e foi realizado com apenas dois usuários, então nada se pode afirmar quanto a efeitos de seu uso prolongado. Um ponto limitante do rastreamento de movimento foi o fato do Google Cardboard não possuir um rastreamento da posição no espaço, apenas da direção e orientação. Essa limitação faz com que os movimentos na direção vertical, não sejam sentidos, causando uma sensação de estranhamento, e os movimentos laterais ou desvios durante a caminhada foram comuns durante os testes realizados, fazendo com que o usuário pisasse para fora da esteira. Este problema também poderia ser mitigado com o uso de uma esteira com lona mais larga.

O modelo de *smartphone* utilizado nos testes foi o Moto G5, da Motorola. Rodando o aplicativo de realidade virtual nele, obteve-se uma *frame rate* de 60 fps a maior parte do tempo.

6 Conclusões e Trabalhos Futuros

Considerando os resultados mostrados no capítulo anterior, conclui-se que o sistema de controle funcionou bem, alcançando os objetivos propostos no trabalho. O usuário pôde variar sua velocidade, mantendo-se próximo do centro da esteira. A saída do controle se tornou muito variável, estando muitas vezes com valores nos limites superior ou inferior, por conta do alto ganho da parcela derivativa do controlador e do ruído presente na leitura de posição realizada pelo sensor ultrassônico, apesar da filtragem. Esta variância não representou um problema por ser filtrada pela dinâmica lenta do motor, mas um melhor ajuste do controle ou redução da variabilidade das leituras feitas pelo sensor são trabalhos futuros para a melhora do sistema.

A realidade virtual criada para o projeto também se comportou como esperado, sendo que a imersão foi prejudicada pela falta de rastreamento da posição da cabeça na direção vertical e dos movimentos laterais do usuário. Outros sistemas de realidade virtual, como o *Oculus Rift* e *HTC Vive* possuem este tipo de sistema, que seria também uma boa opção para substituir o sensor ultrassônico, que possui muitos erros de medição.

Foi observada a facilidade do usuário em caminhar com pequenos desvios da direção da esteira, e esta limitação também poderia ser remediada por um rastreamento de posição. Isso pode ocorrer pela falta de realimentação visual, já que dentro da realidade virtual este desvio não é mostrado. Outra solução para estes desvios do usuário seria a criação de um sistema de esteiras bidirecional, como o Infinadeck, permitindo ao usuário caminhar em qualquer direção. Além destas adaptações para outros sistemas de realidade virtual e grandes modificações no projeto, algumas mudanças simples ao ambiente virtual podem ser trabalhos futuros para melhora da experiência do usuário, entre eles: a adição de mais elementos e variabilidade ao ambiente criado, a adição de mais estatísticas ao painel, como calorias gastas e tempo de corrida, um sistema para salvar e analisar as corridas feitas, a criação de outros ambientes para caminhada ou a adaptação do sistema para uso como controle de outros jogos em realidade virtual.

Uma esteira de maior potência também poderia ser utilizada para obter uma dinâmica mais rápida e para poder alcançar velocidades mais altas. O aumento de velocidade máxima também teria de vir aliado a um aumento na segurança do usuário, para evitar quedas, como o uso de um arnês e a correção do problema do desvio no caminhar. Outra restrição, também imposta pela esteira, foi a imposição de uma velocidade mínima após a arrancada da esteira. Esta limitação ocorreu por conta do circuito de potência não suportar que a esteira seja ligada e desligada muitas vezes seguidas, mas fez com que fossem necessários os botões virtuais de iniciação e parada da esteira.

Apesar destas limitações, mostrou-se que um sistema de movimentação para realidade virtual baseado numa esteira ergométrica é possível, construindo-se um protótipo que realizava esta função. Ao longo do projeto, aprendizados de diversas áreas do curso de Engenharia de Controle e Automação, como a programação, uso de microcontroladores, modelagem de sistemas, teoria de controle, acionamento de máquinas elétricas e instrumentação foram necessários.

7 Referências

Aldaba, C. N.; White, P. J.; Byagowi, A. e Moussavi, Z., "Virtual reality body motion induced navigational controllers and their effects on simulator sickness and pathfinding", 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, pp. 4175-4178, 2017.

Athletic, "Esteira Trainee 4EE". Manaus: Universal Fitness da Amazônia LTDA, 2011, 20 pg.

Bazanella, A. S., Gomes da Silva Jr., J.M., "Sistemas de Controle: princípios e métodos de projeto", Porto Alegre: Editora UFRGS, 2005. 306 p.

Coz, D., Plagemann, C. e Smus, B., "Cardboard: VR for Android", 2014. Disponível em: <https://www.google.com/events/io/schedule/session/603fe228-89c5-e311-b297-00155d5066d7>. Acesso em: 26/10/2017.

Google, "It's your turn to make it", 2017. Disponível em: <vr.google.com/cardboard/manufacturers/>. Acesso em: 18/10/2017.

Infinadeck, "Press Release", 2016. Disponível em: <http://www.infinadeck.com/press-release/>. Acesso em: 19/10/2017.

Latta, J.N., Oberg, D.J., "A Conceptual Virtual Reality Model". IEEE Computer Graphics & Applications, Vol. 14, num. 1: pp 23-29, Jan. 1994.

Machado, L. S., "Conceitos Básicos Da Realidade Virtual". INPE, São José dos Campos, 1995.

Medina, E., Fruland, R., e Weghorst, S., "Virtusphere: Walking in a Human Size VR "Hamster Ball"", HIT Lab, University of Washington, Seattle, Washington, EUA, 2008.

Nakatani, A. M., Guimarães, A. V., Neto, V. M., "Medição Com Sensor Ultrassônico HC-SR04", UTFPR, Curitiba, 2014.

Nichols, S., Patel, H., "Health and safety implications of virtual reality: a review of empirical evidence". Applied Ergonomics, vol. 33, num. 3, 2002.

ON Semiconductor, "2N3903, 2N3904 Datasheet: General Purpose Transistors", Denver, Colorado, USA, 2000.

Pierce, D., "Inside Google's Plan To Make Vr Amazing For Absolutely, Positively Everyone", 2016. Disponível em: <www.wired.com/2016/04/google-vr-clay-bavor/>. Acesso em: 18/10/2017.

Prohl, I., Nelson, J., "Handbook of Contenporary Japanese Religions". Brill Handboks on Contemporary Religion, Vol. 6, 2012.

Scheinerman, M., "Exploring Augmented Reality". Haverford College, Dept. of Computer Science, Haverford, Pennsylvania, EUA, 2009.

Unity Technologies, “Unity User Manual”, 2017. Disponível em: <https://docs.unity3d.com/Manual/>. Acesso em: 26/10/2017.

Wemos, “D1: An Arduino UNO Compatible wifi board based no ESP8266EX.”. 2017. Disponível em: <https://wiki.wemos.cc/products:d1:d1>. Acesso em: 25/10/2017.

Zanbaka, C., Babu, S., Xiao, D., Ulinski, A., Hodges, L. F., e Lok, B., "Effects of travel technique on cognition in virtual environments,". *IEEE Virtual Reality 2004*, pp. 149-286, 2004.

Apêndices

Na seção de anexo são apresentados os códigos criados para o funcionamento do projeto.

Apêndice A: Programação do Microcontrolador

Abaixo, encontra-se o código completo utilizado no controlador para o funcionamento do projeto.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
#include "Arduino.h"

#define pinTrigger D4
#define pinEcho D5
#define pinOut D3
#define pinSpeed D2

const char* ssid = "GVT-4FC8"; //Nome da rede Wifi
const char* password = "0503001118"; //Senha da rede Wifi

ESP8266WebServer server(80);

int vel = 0;
const int led = 13;

const long interval = 10; //delta T do controlador em milissegundos
unsigned long previousMillis = 0; //tempo do ultimo ciclo do controlador

//Sensor de velocidade
unsigned long previousCicleMillis = 0; //tempo da ultima passagem do ima
pelo sensor
const float perimeter = 0.14; //parâmetro do eixo, em m.
float sensSpeed = 0; //velocidade lida no sensor em m/s.

//Controle

float SetP = 0.5; //Set point 50cm

float P = 0; //Proporcional
float I = 0; //Integral
float D = 0; //Derivativo

float Kp = 326.15; //Ganho do Proporcional
float Ki = 1.1978; //Ganho do Integrativo
float Kd = 13033; //Ganho do Derivativo

float Idelay = 25.6/2/Kp; //Para valor inicial do controle ser 25.6
float Ddelay = 0;

const float Kt = 1023 / 175; //Conversão V para 0-1023

float Error;

float dist = SetP; //Distância (CV)
float distAux = SetP; //distancia anterior para calculo do derivativo

const int movAvSize = 10; //numero de pontos da media móvel
float distVec[movAvSize];
```

```
const float velSom = 0.000343; //Velocidade do som em cm/us

bool andando = false;
const int maxVel = 350;
const int minVel = 150;
const int maxIncVel = 200;

void handleRoot() {
  long curMillis = millis();
  if (curMillis - previousCicleMillis >= 500) { //velocidade mínima de 1
km/h
    sensSpeed = 0; //Velocidade lida no sensor em m/s
  }
  server.send(200, "text/plain", String(vel) + ";" + String(dist) + ";" +
String(sensSpeed));
}

void handleNotFound() {
  digitalWrite(led, 1);
  String message = "File Not Found\n\n";
  message += "URI: ";
  message += server.uri();
  message += "\nMethod: ";
  message += (server.method() == HTTP_GET) ? "GET" : "POST";
  message += "\nArguments: ";
  message += server.args();
  message += "\n";
  for (uint8_t i = 0; i < server.args(); i++) {
    message += " " + server.argName(i) + ": " + server.arg(i) + "\n";
  }
  server.send(404, "text/plain", message);
  digitalWrite(led, 0);
}

void setup(void) {
  pinMode(led, OUTPUT);
  pinMode(pinOut, OUTPUT); //Inicia saída para o transistor
  pinMode(pinTrigger, OUTPUT); //Inicia Trigger do sensor ultrassônico
  pinMode(pinEcho, INPUT); //Inicia Echo do sensor ultrassônico
  pinMode(pinSpeed, INPUT);
  analogWriteFreq(16000); //Ajusta frequência da escrita para 16kHz
  vel = 0;
  analogWrite(pinOut, 1023 - vel);
  digitalWrite(led, 0);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.println("");

  // Wait for connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());

  if (MDNS.begin("esp8266")) {
    Serial.println("MDNS responder started");
  }

  for (int i = 0; i < movAvSize; i++) //Inicia vetor de distâncias
```

```
    distVec[i] = SetP;

  server.on("/", handleRoot);

  server.on("/ON", [](){
    andando = true;

    server.send(200, "text/plain", "Andando");
  });

  server.on("/OFF", [](){
    andando = false;

    Idelay = 25.6/2/Kp; //Para valor inicial do controle ser 25.6
    Ddelay = 0;
    vel = 0;
    for (int i = 0; i < movAvSize; i++) //Inicia vetor de distâncias
      distVec[i] = SetP;

    dist = SetP; //Distancia (CV)
    distAux = SetP; //distância anterior para calculo do derivativo

    analogWrite(pinOut,1023);
    server.send(200, "text/plain", "Parado");
  });

  server.onNotFound(handleNotFound);

  server.begin();
  Serial.println("HTTP server started");
  //Escrita para salvar resultados em testes
  Serial.print("DistanciaNãoFiltrada;Tempo; Distancia; OUT; P; I; D;
SensSpeed \n");

  attachInterrupt(pinSpeed, speedInterrupt, FALLING); //Interrupção da
leitura de velocidade

}

long tempoUltrassom() {

  digitalWrite(pinTrigger, LOW);
  delayMicroseconds(2);
  digitalWrite(pinTrigger , HIGH);
  delayMicroseconds(10);
  digitalWrite(pinTrigger , LOW);
  return pulseIn(pinEcho, HIGH);

}

void speedInterrupt() {
  long curMillis = millis();
  if (curMillis - previousCicleMillis >= 63) { //velocidade máxima de 8
km/h (2.22 m/s)
    sensSpeed = perimeter * 1000 / abs(curMillis - previousCicleMillis);
//Velocidade lida no sensor em m/s
    previousCicleMillis = curMillis;
  }
}

void loop(void) {
  server.handleClient();

  unsigned long currentMillis = millis();
```

```
  float T = abs(currentMillis - previousMillis);

  if (T >= interval && andando) { //Só realiza as leituras e cálculos a
cada 'interval' milissegundos

    //Lê distancia
    long microsec = tempoUltrassom();
    float newread = microsec * velSom / 2;
    Serial.print(String(newread)+"; ");
    if(newread-dist>(3.0*interval/1000)){ //se a velocidade máxima é 2.22
m/s, a distância não pode variar mais de 2.22*interval
      newread=dist+(3.0*interval/1000);
    }

    if (newread <= 1.10)
    {
      dist = 0;
      for (int i = movAvSize - 1; i > 0; i--)
      {
        distVec[i] = distVec[i - 1];
        dist += distVec[i];
      }

      distVec[0] = newread;
      dist += distVec[0];
      dist = dist / movAvSize;
    }

    //Controle
    Error = SetP - dist;

    T = T/1000; //de ms para s

    //Proporcional
    P = Kp * Error;

    //Integral
    I = Ki*(2*Idelay+Error); //Ki*(Idelay+Error+Idelay);
    Idelay = Error+Idelay;

    //Derivativo
    D = Kd*(-1.2*Ddelay+Error); //Kd*(-Ddelay+Error-Ddelay/5);
    Ddelay = Error-Ddelay/5;


    int proxVel = (int)(Kt*(P + I + D)); //Sinal de controle convertido
para 0-1023

    if (proxVel > maxVel) //Anti-Windup
      Idelay = Idelay-Error;
    else if (proxVel < minVel)
      Idelay = Idelay-Error;

    if (vel == 0 && proxVel > minVel) //Salta de 0 para valor mínimo
      proxVel = minVel;
    else if (proxVel - vel > maxIncVel)//Limita aumento de velocidade
      proxVel = vel + maxIncVel;

    //Serial.println(String(proxVel));
    if (proxVel > maxVel) //Limita vel
      proxVel = maxVel;
    else if (proxVel < minVel)
      proxVel = minVel;
```

```
    vel = proxVel;
    distAux = dist;
    analogWrite(pinOut, 1023 - vel);
    // Escreve na porta serial (para debug)

    Serial.print(String(currentMillis)+"; "+String(dist)+";
"+String(vel)+"; "+String(P)+"; "+String(I)+"; "+String(D)+";
"+String(sensSpeed)+"\n");
    previousMillis = currentMillis;
  }
}
```

Apêndice B: Programação de *Scripts* para Unity

Abaixo, estão alguns dos códigos dos *scripts* criados para o ambiente virtual em Unity.

CenarySpeed.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

public class CenarySpeed : MonoBehaviour {
    public Vector3 velocidade;
    public bool andando = false;
    public GameObject GoButton;
    public GameObject StopButton;
    bool VelDaRede = true;

    Rigidbody rb;
    void Start(){
        rb = GetComponent<Rigidbody> ();
        if(VelDaRede)
            InvokeRepeating("GetTextFun", 0f, 0.1f);
    }
    // Update is called once per frame
    void FixedUpdate () {
        if (andando & !VelDaRede) {
            velocidade.z = velocidade.z + Input.GetAxis ("Vertical");
            if (velocidade.z < 0)
                velocidade.z = 0;
            rb.velocity = -velocidade;
        }
    }
    public void Go(){ // Chamada pelo acionamento do botão Go!
        Debug.Log ("Go!");

        StartCoroutine (StartTreadmill ());

        velocidade = Vector3.zero;
        rb.velocity = velocidade;
        andando = true;
        StopButton.SetActive (true);
        GoButton.SetActive (false);

    }
    public void Stop(){ // Chamada pelo acionamento do botão Stop
        Debug.Log ("Stop!");
```

```
        StartCoroutine (StopTreadmill ());

        velocidade = Vector3.zero;
        rb.velocity = velocidade;
        andando = false;
        StopButton.SetActive (false);
        GoButton.SetActive (true);
    }

    void GetTextFun() {
        StartCoroutine (GetText ());
    }
    IEnumerator GetText() { //Pega Velocidade do Microcontrolador
        WWW www = new WWW("http://192.168.25.60");
        yield return www;
        if (www.error!= null) {
            Debug.Log(www.error);
        }
        else {
            Debug.Log(www.text);
            string[] array = www.text.Split(';');
            if (andando) {
                velocidade.z = -1*float.Parse(array[2]);
                rb.velocity = velocidade;
            }
        }
    }
    IEnumerator StartTreadmill() {
        //Liga Esteira
        WWW www = new WWW("http://192.168.25.60/ON");
        yield return www;
        if (www.error!= null) {
            Debug.Log (www.error);
        } else {
            Debug.Log ("Esteira Andano");
        }
    }
    IEnumerator StopTreadmill() {
        //Desliga Esteira
        WWW www = new WWW("http://192.168.25.60/OFF");
        yield return www;
        if (www.error!= null) {
            Debug.Log (www.error);
        } else {
            Debug.Log ("Esteira Parada");
        }
    }
}
```

SpawnCenary.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpawnCenary : MonoBehaviour {
    public GameObject tree;
    public float periodoMin;
    public float periodoMax;
    public GameObject parentCenary;
```

```
    // Use this for initialization
    Rigidbody rb;
    void Start () {
        rb = GetComponent<Rigidbody> ();
        StartCoroutine (SpawnTrees ());
    }

    IEnumerator SpawnTrees(){
        Quaternion spawnRotation = new Quaternion();
        while (true) {
            if (rb.velocity.z != 0) {
                Vector3 spawnPosition = new Vector3 (Random.Range (15, 50), 0,
100);
                spawnRotation = Quaternion.Euler (0, Random.Range (-
180, 180), 0);
                Instantiate (tree, spawnPosition, spawnRotation, parentCenary.t
ransform);
                spawnPosition = new Vector3 (Random.Range (-50, -15), 0, 100);
                spawnRotation = Quaternion.Euler (0, Random.Range (-
180, 180), 0);
                Instantiate (tree, spawnPosition, spawnRotation, parentCenary.t
ransform);
                yield return new WaitForSeconds (Random.Range (periodoMin / (-
1 * rb.velocity.z), periodoMax / (-1 * rb.velocity.z)));
            } else {
                yield return new WaitForSeconds (Random.Range (periodoMin, peri
odoMax));
            }
        }
    }
}
```

RenderStats.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RenderStats : MonoBehaviour {

    public TextMesh Velocidade;
    public TextMesh Distancia;
    public GameObject Cenario;

    // Update is called once per frame
    void Update () { // Mostra as informações no painel de estatísticas
        Velocidade.text = (-
1*Cenario.GetComponent<CenarySpeed>().velocidade.z*3.6).ToString("0.0") +  " km
/h";
        Distancia.text = (-
1*Cenario.transform.position.z/1000).ToString("0.00") +  " km";
    }
}
```

DestroyOnExit.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class DestroyOnExit : MonoBehaviour {
    void OnTriggerExit(Collider other)
    {
        if (other.tag == "Destructable"){
            Destroy(other.gameObject);
        }
        if (other.tag == "Floor"){
            Vector3 initpos = new Vector3 (0, 0, 400);
            other.transform.position = other.transform.position + initpos;
        }
    }
}
```